8-2020

A General Approach to Lifting-Line Theory, Applied to Wings with Sweep

Jackson T. Reid
Utah State University

A GENERAL APPROACH TO LIFTING-LINE THEORY,

APPLIED TO WINGS WITH SWEEP

by

Jackson T. Reid

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Aerospace Engineering

Approved:

Douglas F. Hunsaker, Ph.D. Major Professor	Andreas Malmendier, Ph.D. Committee Member
Geordie Richards, Ph.D. Committee Member	Barton L. Smith, Ph.D. Committee Member
Zhongquan C. Zheng, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

A General Approach to Lifting-Line Theory,

Applied to Wings with Sweep

by

Jackson T. Reid, Doctor of Philosophy

Utah State University, 2020

Major Professor: Douglas F. Hunsaker, Ph.D.
Department: Mechanical and Aerospace Engineering

Implementations of lifting-line theory predict the lift of a finite wing using a sheet of semi-infinite vortices extending from a vortex filament placed along the locus of aerodynamic centers of the wing. Prandtl's classical implementation is restricted to straight wings in flows without sideslip. In this work, it is shown that lifting-line theory can be extended to swept wings if, at the control points where induced velocity is calculated, the second derivative of the locus of aerodynamic centers is zero and the trailing vortices are perpendicular to the locus of aerodynamic centers. Therefore, a general implementation of lifting-line theory is presented that conditionally forces the second derivative of the locus of aerodynamic centers to zero at each control point and joints each trailing vortex such that there is a finite segment of the trailing vortex that lies perpendicular to the locus of aerodynamic centers. Consideration is given to modeling the locus of aerodynamic centers and section aerodynamic properties of swept wings. The resulting general formulation is analyzed to determine its sensitivity to closure parameters, accuracy, and numerical convergence.

(196 pages)

PUBLIC ABSTRACT

A General Approach to Lifting-Line Theory,
Applied to Wings with Sweep
Jackson T. Reid

Lifting-line theory is one simple method of predicting the lift produced by a wing. The traditional implementation of lifting-line theory, developed in 1918, is limited to predicting the lift of traditional straight wings. In this work, lifting-line theory is extended to predict the lift produced by modern swept (or "v-shaped") wings by strategically handling the singularities inherent to the theory. The resulting formulation is shown to be both accurate and computationally inexpensive, when compared to experimental and higher-fidelity computational results, demonstrating the method's usefulness as an aerodynamic design tool. Because of the low computational cost and accuracy of the method described in this work, it is of interest to a range of persons involved in the design, flight, and control of aircraft. The method can be used in large design-space studies, for which high-fidelity aerodynamic tools are computationally prohibitive, and in real-time applications, such as flight simulation and aircraft control systems.

The most important equation:

$$J + M + m + j = \infty$$

ACKNOWLEDGMENTS

The author would like to thank the Office of Research and Graduate Studies at Utah State University, the Mechanical & Aerospace Engineering Department, and Assistant Professor Douglas Hunsaker for their support through the Presidential Doctoral Research Fellowship (PDRF) program. Additional funding for this work was provided by the U.S. Office of Naval Research Sea-Based Aviation program (Grant No. N00014-18-1-2502) with Brian Holm-Hansen as the program officer.

In addition, the author would like to thank Associate Professor Andreas Malmendier for his contribution to the analysis of the parabolic vortex segment described in Chapter 3. In particular, the mathematical magic performed in Section 3.2.

Finally, the author would like to thank Jeffery Taylor for his welcome feedback and tireless review throughout all stages of this work.

Jackson T. Reid

CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

While not imperative to the ideas presented in the body of this work, the author would not consider this body of research complete if he were not able to provide the reader with at least the most basic physical intuition into why lift is generated. Over a century has elapsed since mankind mechanically achieved sustainable, controllable flight. However, the reason for why lift is generated by an airfoil (the cross-sectional shape of a wing) is still largely misunderstood. Therefore, this work is prefaced with a presentation of the elegant nature of lift. As a result, the reader should receive an adequate lens through which to appreciate the collection of work presented thereafter.

A Slow Start

Consider a solid body in a very slow fluid flow, often referred to as a "creeping" flow. In such a flow, the inertia of the fluid (i.e. its resistance to changes in speed or direction) is negligible when compared to its viscosity (i.e. its internal friction). The flow over the cylinder depicted in Fig. 1a, and the flow over the airfoil depicted in Fig. 2a, are examples of solid bodies in this type of creeping flow. In both cases, the fluid is pushed out from in front of the solid body, then "fills in" the area behind the body. Because the inertia of the fluid is negligible, the fluid changes direction with ease and follows the body all the way around until it meets itself on the other side.

The changes in the speed and direction of the flow in Figs. 1a and 2a correlate to differences in pressure within the fluid. In general, such differences in pressure occur in a flow when the motion of that flow is resisted or accelerated in some way—by a solid body, by the inertia or viscosity of the fluid itself, or by some other force (e.g. gravity). It can be said that the motion of a fluid is a balance between the pressure differences in the flow and the fluid's inertia and viscosity. This balance typically results in lower pressure where the flow is faster, and higher pressure where the flow is slower.

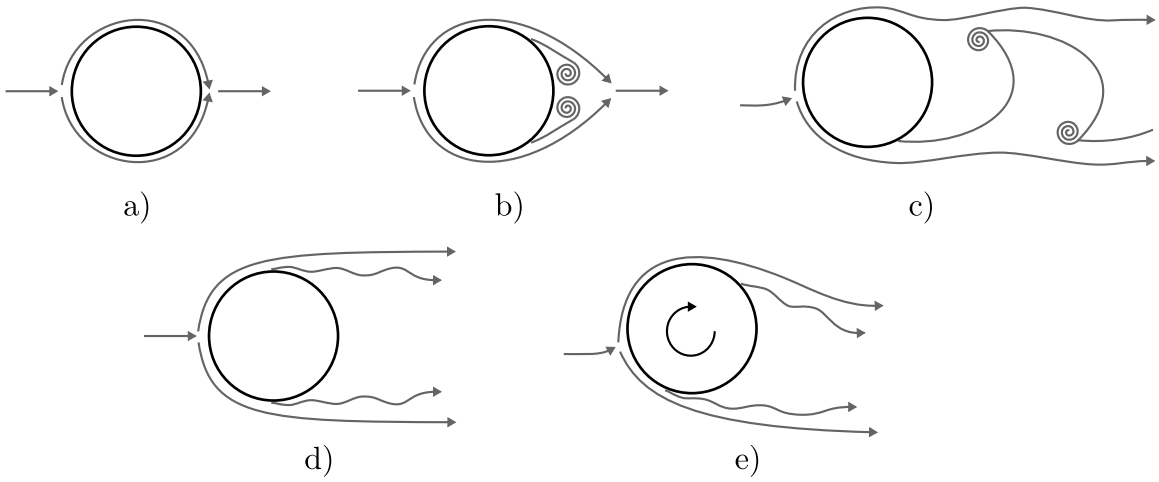

Fig. 1: Flow over a cylinder.

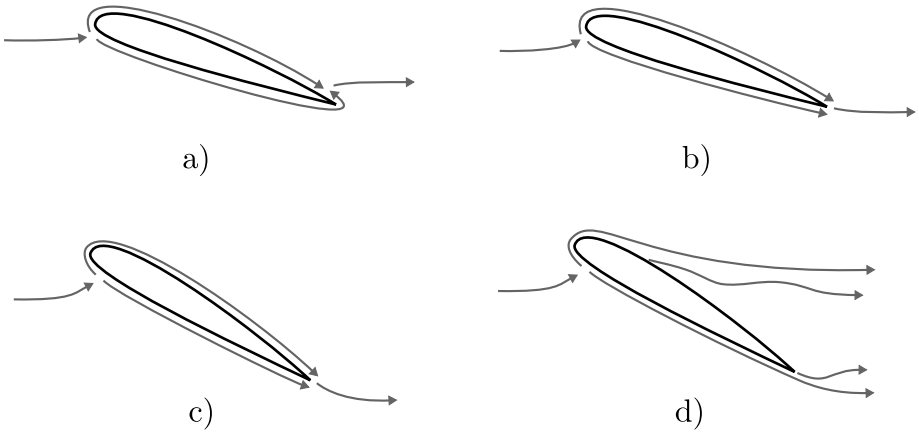

Fig. 2: Flow over an airfoil.

The pressure differences in the creeping flow around the solid bodies in Figs. 1a and 2a result in a force on the surface of each body. However, notice the symmetry of the creeping flow over the cylinder in Fig. 1a. Because the flow over the top and bottom of the cylinder is symmetric, there is no net force exerted vertically on the cylinder (i.e. no lift). The effect is the same for the creeping flow over the airfoil in Fig. 2a, though more difficult to visualize. In creeping flow, neither the cylinder nor the airfoil produce lift because the symmetric pressures that correspond to the flow do not exert a net force in the vertical direction.

Building Inertia

Now, consider a flow that is traveling just fast enough that its inertia is no longer negligible. When this happens, the flow is more resistant to changing speed or direction. In this regime, the flow over the cylinder shown in Fig. 1b cannot turn completely to fill in behind the cylinder. Instead, there is a recirculating region behind the cylinder. Still, the flow is symmetric above and below the cylinder in Fig. 1b, and no lift is produced.

If the flow moves a little faster still, the recirculation region behind the cylinder becomes unstable. Small differences in the flow above and below the cylinder cause the recirculation region to oscillate, as depicted in Fig. 1c. As the amount of fluid flowing over the top and bottom of the cylinder oscillates back and forth, the pressures on the top and bottom of the cylinder oscillate opposite of one another. The resulting pressure asymmetry results in a force in the vertical direction: Lift!

The lift produced by the flow depicted in Fig. 1c is less than ideal. The vertical force acting on the cylinder oscillates between pushing up and pushing down. If the flow speed increases further, the periodic oscillation of the flow behind the cylinder becomes increasingly chaotic until it forms a turbulent wake, depicted in Fig. 1d. Once the turbulent wake is formed, the amount of fluid flowing above and below the cylinder returns to symmetry, and the corresponding pressure distribution again results in no lift.

Gaining Control

The lift produced by the flow around a cylinder is seen to be erratic. It exists only for certain flow speeds, and when it does exist, it oscillates between pushing the cylinder up and pushing it down. How, then, can lift generation be controlled? For a cylinder, one method of creating a consistent asymmetry in the flow is by spinning the cylinder, as shown in Fig. 1e. The difference in the speed of the flow relative to the cylinder's top and bottom surfaces results in relatively more fluid flowing over the top of the cylinder and a corresponding asymmetry in the pressure, generating a sustained amount of lift. However, a spinning cylinder is not a very feasible nor efficient means of generating lift.

A more-elegant approach to sustained lift is obtained by modifying the geometry of the body in the flow. Consider the flow over the airfoil in which the flow's inertia is not negligible, depicted in Fig. 2b. As was the case with the cylinder, the inertia of the flow resists changes in direction, and the flow below the airfoil does not turn completely around the sharp corner at the end of the airfoil to fill in behind the airfoil. Conversely, because of the more gradual curve at the front of the airfoil, the flow over the top surface does turn to the backside of the airfoil. Now, unlike the creeping flow shown in Fig. 2a, the flow over the top of the airfoil does not meet the flow from beneath at the back of the airfoil, but continues to the end of the airfoil. The result is an asymmetric flow, in which, as compared with the flow in Fig. 2a, relatively more fluid is flowing over the top of the airfoil and less is flowing along the bottom. The accompanying asymmetric pressure distribution—with higher pressure on the bottom and lower pressure on top—results in lift!

It should now be clear why airfoils are shaped the way they are. They are designed to produce a consistent, controllable amount of lift by forcing the flow beneath to separate at the sharp end of the airfoil and by allowing the top flow to turn along the more-gradually curved surface at the front, the net result being an asymmetric pressure distribution in the vertical direction.

The amount of force generated by the airfoil depends on the amount of asymmetry in the pressure associated with the flow. Increasing the angle at which the airfoil encounters the flow, in Fig. 2c for example, increases the pressure asymmetry and results in more lift. However, if the angle of the airfoil is increased too much, the flow over the top no longer turns around to the back of the airfoil, see Fig. 2d, and a wake region similar to the cylinder's in Fig. 1d-e forms, decreasing the lift and increasing the drag. This is known as "stall".

Under Pressure

It can be understood that lift can occur for any solid body, so long as the pressure distribution is asymmetric in the vertical direction. The pressure asymmetry experienced by the body can be expressed as the mathematical quantity known as *circulation*. With-

out resorting to mathematical equations, circulation is best envisioned by the flow over a rotating cylinder depicted in Fig. 1e, and the lifting airfoils depicted in Figs. 2b-c. In these flows—whose vertically asymmetric pressure distribution results in lift in the upward direction—the circulation is the idea that the the speed and quantity of fluid flowing in the clockwise direction (i.e. above the body) is greater than that of the fluid in the counter-clockwise direction (i.e. beneath the body). Likewise, a flow whose vertically asymmetric pressure distribution results in downward lift can be conceptualized as the speed and quantity of fluid flowing in the counter-clockwise direction being greater than that of the fluid flowing in the clockwise direction. Flows that result in zero lift, such those is Fig. 1a and Fig. 2a, demonstrate a balance of the flow in the clockwise and counter-clockwise directions.

The concept of circulation is a very important mathematical tool that is used to model the lift on a body in certain types of flow. The focus of this work is the examination and analysis of several such mathematical models. Herein, the lift over a wing is predicted by strategically representing the wing as a distribution of circulation, in the form of vortices. At first, the amount of circulation is not known, but through the various means leveraged in the body of this work, the circulation is determined, resulting in a prediction of the lift produced by the wing.

With this in mind, please enjoy the following dissertation.

Jackson T. Reid

Further reading: If the reader wishes to study further, the following references provide insightful material about the physical phenomena of lift:

- *Fluid Mechanics* by P. K. Kundu, I. M. Cohen, and D. R. Dowling
  (5^{th} edition; Academic Press; 2012; pgs. 388-399 and 696-701)
- *Understanding Aerodynamics* by Doug McLean
  (John Wiley & Sons, Ltd.; 2013; pgs. 31-33, 163-168 and 259-302)

CHAPTER 1

INTRODUCTION

Lifting-line theory is based on the conjecture made in the study of potential flow that the physical flow around a body can be represented as the velocity field induced by a distribution of vortices added to a freestream [1–6]. In particular, the assertion is made in lifting-line theory that the flow over a finite, high-aspect-ratio wing can be represented by a sheet of semi-infinite vortices extending from a single, variable strength vortex filament, placed along the locus of aerodynamic centers of the wing, as depicted in Fig. 1.1. The circulation strength of the vortex sheet, as a function of spanwise location, is equivalent to the change in the circulation of the bound vortex filament. The crux of lifting-line theory is the determination of the circulation distribution that results in a field of induced velocity along the wing's locus of aerodynamic centers that results in the same spanwise lift distribution as the corresponding physical wing, for a given freestream condition.

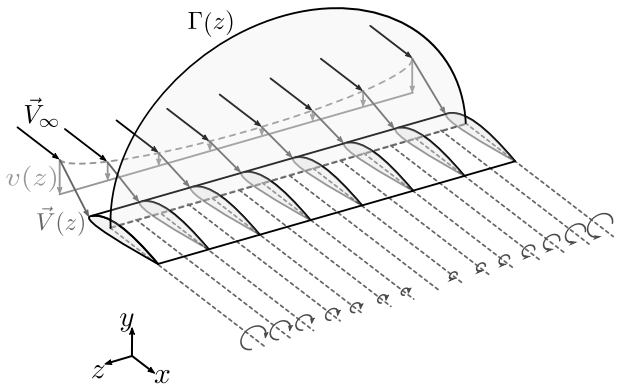

Fig. 1.1: Depiction of Prandtl's classical lifting-line theory.

1.1 Prandtl's Classic Implementation

In Prandtl's classical derivation of lifting-line theory [1–3], the circulation distribution, $\Gamma(z)$, is found by equating two definitions for the section lift coefficient, $\tilde{C}_L$. First, from the Kutta-Joukowski law [7, 8], the section coefficient of lift is given by

$$\tilde{C}_L = \frac{2\Gamma}{V_\infty c} \tag{1.1}$$

where V_∞ is the magnitude of the freestream velocity, and c is the chord length of the section. Second, it is assumed that the section lift coefficient is a linear function of angle of attack

$$\tilde{C}_L = \tilde{C}_{L,\alpha}\big(\alpha_{\text{eff}} - \alpha_{L0}\big) \tag{1.2}$$

where $\tilde{C}_{L,\alpha}$ and α_{L0} are properties of the section airfoil, and α_{eff} is the effective angle of attack. The effective angle of attack deviates from the global angle of attack of the wing, α, due to the velocity that the semi-infinite vortex sheet induces along the locus of aerodynamic centers. Assuming that the velocity induced by the vortex sheet is small compared to the freestream, the effective angle of attack is

$$\alpha_{\text{eff}}(z_0) = \alpha - \frac{1}{4\pi V_\infty}\int_{-b/2}^{b/2} \frac{(d\Gamma/dz)}{z_0 - z} dz \tag{1.3}$$

where z_0 is a spanwise location along the wing (see Fig. 1.1). Note that, in the case of a wing without sweep or dihedral, the locus of aerodynamic centers is assumed to lie along a straight line, and therefore the bound vortex filament does not have any influence on the effective angle of attack. Equating Eqs. (1.1) and (1.2), in combination with Eq. (1.3), results in Prandtl's fundamental lifting-line equation for a given section, z_0, along the wing [1–3]

$$\alpha(z_0) = \frac{2\Gamma(z_0)}{V_\infty c(z_0)\tilde{C}_{L,\alpha}(z_0)} + \alpha_{L0}(z_0) + \frac{1}{4\pi V_\infty}\int_{-b/2}^{b/2} \frac{(d\Gamma/dz)}{z_0 - z} dz \tag{1.4}$$

where the only unknown is the circulation distribution, $\Gamma(z)$.

In order to find a lift distribution that satisfies Eq. (1.4), consider the change of variables

$$z = -\frac{b}{2}\cos(\theta) \tag{1.5}$$

$$dz = \frac{b}{2}\sin(\theta)d\theta \tag{1.6}$$

With this transformation of variables, the general circulation distribution, $\Gamma(z)$, can be expressed in terms of θ as the Fourier sine series

$$\Gamma(\theta) = 2bV_\infty \sum_{n=1}^{\infty} A_n \sin(n\theta) \tag{1.7}$$

Differentiating Eq. (1.7) with respect to the spanwise coordinate, z, gives

$$\Gamma'(z) = \frac{d\Gamma}{dz} = \frac{d\Gamma}{d\theta}\frac{d\theta}{dz} = 2bV_\infty \sum_{n=1}^{\infty} nA_n \cos(n\theta)\frac{d\theta}{dz} \tag{1.8}$$

The required Fourier coefficients, A_n, are found by satisfying Eq. (1.4) at N locations along the wing. The solution to the resulting linear system of equations provides the first N coefficients of the circulation distribution, $\Gamma(\theta)$. Thus, Eq. (1.4) can be rewritten as

$$\begin{aligned}\alpha(\theta_0) &= \frac{4b}{c(\theta_0)\tilde{C}_{L,\alpha}(\theta_0)} \sum_{n=1}^{N} A_n \sin(n\theta_0) + \alpha_{L0}(\theta_0) + \frac{1}{\pi}\int_0^\pi \frac{\sum_{n=1}^{N} nA_n \cos(n\theta_0)}{\cos(\theta) - \cos(\theta_0)} d\theta \\ &= \frac{4b}{c(\theta_0)\tilde{C}_{L,\alpha}(\theta_0)} \sum_{n=1}^{N} A_n \sin(n\theta_0) + \alpha_{L0}(\theta_0) + \sum_{n=1}^{N} nA_n \frac{\sin(n\theta_0)}{\sin(\theta_0)}\end{aligned} \tag{1.9}$$

for each of the N sections along the wing.

Though lifting-line theory can provide valuable aerodynamic intuition, it is traditionally limited to high-aspect-ratio wings in steady, incompressible flows with a small angle of attack. In addition, Prandtl's implementation does not provide solutions for non-straight wings or wings in sideslip. The difficulty faced when attempting to extend Prandtl's implementation to wings with sweep is the occurrence of infinite, self-induced velocities at the control points along the wing. The nature of these infinite velocities is discussed in Chap-

ter 2. Several implementations have been developed in an attempt to extend lifting-line theory to wings with sweep, though each with its own drawbacks.

1.2 Vortex Core and Integral Cutoff Implementations

In several implementations of lifting-line theory, a finite core or an integral cutoff is applied to the vortex models [9–12]. This approach removes the portions of the velocity field responsible for infinite, self-induced velocities. For example, consider the velocity induced by the straight vortex segment shown in Fig. 1.2

$$\vec{V} = \frac{(|\vec{r}_a| + |\vec{r}_b|)(\vec{r}_a \times \vec{r}_b)}{|\vec{r}_a||\vec{r}_b|(|\vec{r}_a||\vec{r}_b| + \vec{r}_a \cdot \vec{r}_b)} \tag{1.10}$$

The velocity induced by the vortex tends to infinity as the distance from the vortex, $(|\vec{r}_a||\vec{r}_b| + \vec{r}_a \cdot \vec{r}_b)$, approaches zero. If a finite-core is added to Eq. 1.10, the vortex segment no longer induces infinite velocities, as shown by the model [12]

$$\vec{V} = \frac{d^2}{(r_c^4 + d^4)^{1/2}} \frac{(|\vec{r}_a| + |\vec{r}_b|)(\vec{r}_a \times \vec{r}_b)}{|\vec{r}_a||\vec{r}_b|(|\vec{r}_a||\vec{r}_b| + \vec{r}_a \cdot \vec{r}_b)} \tag{1.11}$$

where d is either the distance of $\vec{P}$ normal to the vortex segment, if the projection of $\vec{P}$ falls on the segment, or the distance from $\vec{P}$ to the closest end point (i.e. either $|\vec{r}_a|$ or $|\vec{r}_b|$).

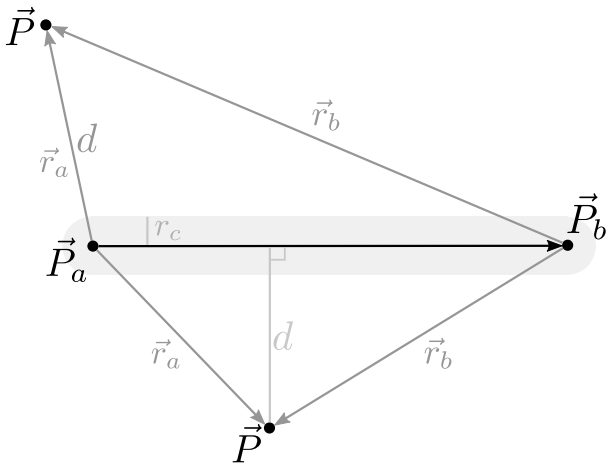

Fig. 1.2: The geometry defining the velocity induced at two example points, $\vec{P}$, by a finite vortex segment with a finite core of radius r_c.

The velocity induced by the vortex described in Eq. 1.11 is sensitive to the radius, r_c, used in applying the finite vortex core, as shown in Fig. 1.3. Furthermore, while these methods can accurately predict the total lift and induced drag of a wing, they are less accurate in predicting how the lift is distributed along the wing span, as will be demonstrated in Chapter 7.

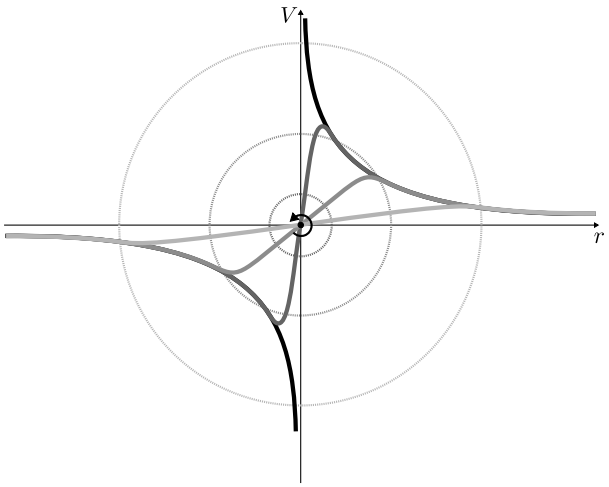

Fig. 1.3: Depiction of the velocity profile induced by several vortices with various finite core radii.

1.3 Weissinger's Implementation

In another commonly used implementation, first developed by Weissinger [13] and applied by others [14–17], the control points at which the induced velocities are calculated are moved off the locus of aerodynamic centers to the *three-quarter chord line* (i.e. the locus of points along a wing at 3/4 the distance from the leading edge to the trailing edge of the wing). Then, instead of relating the induced velocities to a model of the airfoil section properties to determine the circulation distribution, the induced velocity at each control point is required to be tangential to the wing camber line, in accordance to the analytical solution for a flat plate [18]. The benefit of this methodology is that section properties need not be known a priori. However, the downside of this approach is that, because the only influential property of the airfoil section is the slope of its camber line at the three-quarter

chord point, the section properties are limited to thin airfoils with small amounts of camber, see Fig. 1.4.

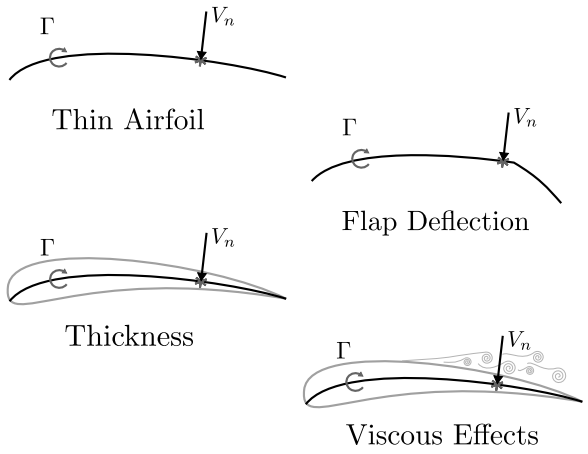

Fig. 1.4: Depiction of four airfoils with the same 3/4 chord camber slope.

1.4 Phillip's Modern Implementation

A numerical analog of Prandtl's lifting-line implementation was developed by Phillips and Snyder [5, 19]. In contrast to the continuous functions used in Prandtl's classical implementation, Phillips's implementation of lifting-line theory separates the bound vortex filament and trailing vortex sheet into a discrete number of abutted *horseshoe vortices*, each consisting of a constant-strength vortex segment and two semi-infinite vortices. Endpoints of the bound portion of each horseshoe vortex lay on the wing's locus of aerodynamic centers, and the trailing portion of each horseshoe vortex is aligned with the freestream. The local velocity is calculated at a control point located on each bound vortex segment with the equation

$$\vec{V}_i = \vec{V}_\infty + \sum_{j=1}^{N} \Gamma_j \vec{v}_{ji} \tag{1.12}$$

where Γ_j is the strength of each horseshoe vortex, and $\vec{v}_{ji}$ is the influence of horseshoe vortex j on control point i.

Similar to Prandtl's implementation, the strength of each horseshoe vortex, Γ_j, is not initially known, but is determined by relating two definitions of the force generated by each section of the wing. Using the vectorized form of the Kutta-Joukowski law [5, 7, 8, 19], the force that each bound vortex exerts, given a local velocity vector, $\vec{V}_i$, can be described by the equation

$$d\vec{F}_i = \rho\Gamma_i\vec{V}_i \times d\vec{l}_i \tag{1.13}$$

This definition for section force is related to information obtained from an analytic, numerical, empirical, or experimental prediction of the section lift coefficient as a function of the local velocity, $\tilde{C}_L(\vec{V}_i)$. Using Eq. (1.12) in Eq. (1.13), and equating the result to the section lift coefficient, gives a non-linear system of equations which can be solved iteratively

$$\rho\Gamma_i\left|\left(\vec{V}_\infty + \sum_{j=1}^{N}\Gamma_j\,\vec{v}_{ji}\right) \times d\vec{l}_i\right| - \frac{1}{2}\rho V_\infty^2\tilde{C}_L(\vec{V}_i)dA_i = 0 \tag{1.14}$$

It is convenient to rewrite Eq. (1.14) in the non-dimensionalized form

$$2\left|\left(\vec{u}_\infty + \sum_{j=1}^{N}G_j\,\vec{v}_{ji}\right) \times \vec{\zeta}_i\right|G_i - \tilde{C}_L(\vec{V}_i) = 0 \tag{1.15}$$

where

$$\vec{u}_\infty = \frac{\vec{V}_\infty}{V_\infty}, \quad G_i = \frac{\Gamma_i}{V_\infty}, \quad \vec{\zeta}_i = \frac{d\vec{l}_i}{dA_i} \tag{1.16}$$

Solving this non-linear system for G_i provides the circulation distribution of the wing. The distributions of lift and induced drag are then found by summing the forces calculated with Eq. (1.13), using the induced velocities resulting from Eq. (1.12).

The advantage of Phillips' implementation lies in the fact that explicit integration is not necessary in Eq. (1.12) to determine induced velocities, as opposed to Eq. (1.3). The trade off is made by taking advantage of modern computational power to solve a non-linear system of equations, Eq. (1.15), in lieu of the linear system that is the basis of Prandtl's implementation, Eq. (1.9). In Chapter 6, this advantage is shown to be leverageable in the derivation of a general implementation of lifting-line theory, allowing for lifting-line theory

to be applied to wings with sweep and wings in sideslip.

Unfortunately, the algorithm developed by Phillips suffers from an inability to numerically grid-converge for wings with sweep or in sideslip. Consider an untapered wing with NACA 0012 airfoil sections, an aspect ratio of 5, and 45° sweep, at an angle of attack of 4.2° and zero sideslip. This configuration is taken from an experiment performed by Weber and Brebner [20] and will be used here, and again in Chapter 7, to assess the accuracy of the lifting-line implementations discussed in this work. Figure 1.5 shows the total lift coefficient, as predicted by Phillips' implementation, as well as the root-mean-square (RMS) change in circulation distribution as the number of nodes along the span increases. Using a straight wing, Phillips suggests that grid convergence is obtained for node counts above $N = 80$ along the span [19]. Notice that the circulation distribution predicted for a swept wing by Phillips' implementation in Fig. 1.5 fails to converge with an increasing node count, even for node counts well above the suggested value.

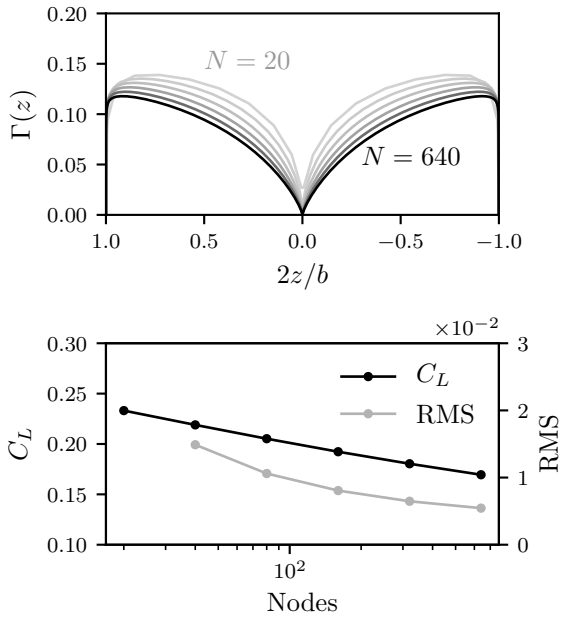

Fig. 1.5: The circulation distribution predicted by Phillips' implementation, for several node counts (top). The RMS change in circulation distribution and the total lift coefficient as a function of the node count (bottom).

For a convergent numerical algorithm, it is expected that the RMS change in the predicted circulation distribution will tend to zero as the number of nodes used in the computation increases. Furthermore, the numerical algorithm's order of convergence is measured by the rate at which the RMS change approaches its fully refined value (i.e. at $N = \infty$), using Richardson extrapolation [21][1]. An algorithm is considered to be convergent if its rate of convergence is at least one, while a convergence rate around two or above is preferable [21]. Considering the wing described above, but with *zero sweep*, Phillips implementation demonstrates a lift coefficient convergence of 1.999, and a RMS convergence of 1.475. In contrast, the results in Fig. 1.5 show that the convergence rate of the RMS change in the circulation distribution predicted by Phillips' implementation is 0.635, and the convergence rate of the total lift coefficient is 0.129. This lack of numerical convergence precludes the application of Phillips' lifting-line implementation to wings with sweep or in sideslip. So, while Phillip's implementation of lifting-line theory is indeed a "modern" approach to Prandtl's lifting-line—in the sense that it takes advantage of modern computing power to solve a non-linear system of equations to remove the need for explicit integration—it does not, in effect, expand upon Prandtl's implementation to allow for the accurate prediction of the lift produced by wings with sweep or wings in sideslip.

In this work, a general implementation of lifting-line theory will be presented, with which the aerodynamic properties of swept wings and wings in sideslip can be predicted, while avoiding the drawbacks of the other implementations discussed above. In particular, the general implementation of lifting-line theory derived in this work permits the use of arbitrary models for section properties, is relatively insensitive to model closure parameters, and numerically grid resolves. It should be noted that, while work has been performed to adapt lifting-line theory for low aspect-ratio wings [4,22,23], that topic will not be addressed herein.

[1]The process of Richardson extrapolation is described in Appendix A.

CHAPTER 2

MODELS FOR INDUCED VELOCITY

The first step in deriving a general implementation of lifting-line theory is to describe the velocity induced along the locus of aerodynamic centers more generally than the description in Eq. (1.3). This can be done by considering the influence of the trailing vortex sheet and the bound vortex filament on a locus of aerodynamic centers described by the general function $f(z)$ in the x-z plane.

2.1 Bound Vortex Filament

The velocity induced by a differential element of the vortex filament at a point along the locus of aerodynamic centers, $[f(z_0), 0, z_0]$, is governed by the Biot-Savart law [5, 24]

$$d\vec{V} = \frac{\Gamma}{4\pi}\frac{d\vec{l}\times\vec{r}}{r^3} \tag{2.1}$$

where the vectors $d\vec{l}$ and $\vec{r}$ are defined as

$$d\vec{l} = -\frac{d}{dz}\Big[f(z), 0, z\Big] = -\Big[f'(z), 0, 1\Big]dz \tag{2.2}$$

$$\vec{r} = \Big[f(z_0) - f(z), 0, z_0 - z\Big] \tag{2.3}$$

and r is the magnitude of $\vec{r}$. The total velocity induced at the point z_0 by the entire bound vortex filament is found through the integration of Eq. (2.1)

$$\vec{V}_{\mathrm{LAC}}(z_0) = \int_{-b/2}^{b/2} \frac{-\Gamma(z)}{4\pi} \frac{\Big[0, \frac{f(z_0)-f(z)}{z_0-z} - f'(z), 0\Big]}{\Big(1 + \Big(\frac{f(z_0)-f(z)}{z_0-z}\Big)^2\Big)^{3/2}(z_0-z)^2}dz \tag{2.4}$$

2.1.1 Numerical Implementation of Bound Vortex Segments

Having described the influence of the bound vortex filament in Eq. (2.4), consider the influence of the finite number of bound vortex segments used in Phillips' implementation to approximate the continuous filament. Using the notation in Fig. 2.1, the velocity induced

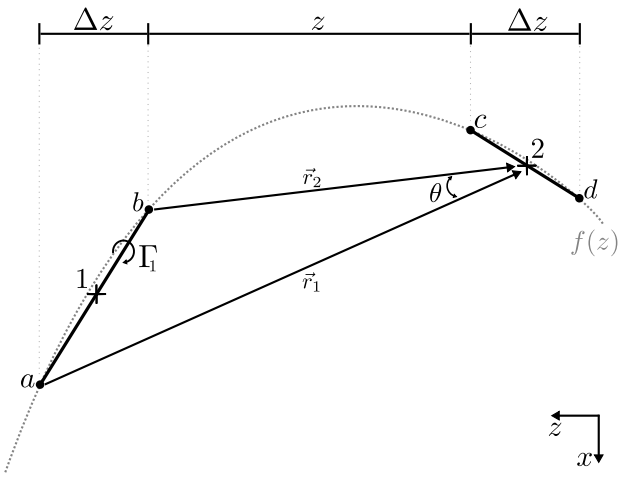

Fig. 2.1: The geometry used to define the velocity induced at a point by a single linear vortex segment. ($\Delta z > 0$ and $z = z_b - z_c$)

at a point by a single linear vortex segment can be described as [5]

$$d\vec{V}_2 = \frac{\Gamma_1}{4\pi} \frac{(r_1 + r_2)(\vec{r}_1 \times \vec{r}_2)}{r_1 r_2 (r_1 r_2 + \vec{r}_1 \cdot \vec{r}_2)} \tag{2.5}$$

where r_1 and r_2 are the magnitudes of $\vec{r}_1$ and $\vec{r}_2$. Using point b as a reference, the vectors $\vec{r}_1$ and $\vec{r}_2$ are defined as

$$\vec{r}_1 = \Big[\frac{f(z_c) + f(z_c - \Delta z)}{2} - f(z_b + \Delta z), 0\,, -z - \frac{3\Delta z}{2} \Big] \tag{2.6}$$

$$\vec{r}_2 = \Big[\frac{f(z_c) + f(z_c - \Delta z)}{2} - f(z_b), 0\,, -z - \frac{\Delta z}{2} \Big] \tag{2.7}$$

As $\Delta z \to 0$, Eq. (2.5) becomes

$$\lim_{\Delta z \to 0} d\vec{V}_2 = \frac{-\Gamma(z_b)}{4\pi} \frac{\left[0, \frac{f(z_c)-f(z_b)}{z} - f'(z_b), 0\right]}{\left(1 + \left(\frac{f(z_c)-f(z_b)}{z}\right)^2\right)^{3/2} z^2} dz \tag{2.8}$$

which is the same as the velocity induced by the continuous bound vortex seen in Eq. (2.4). Thus, as the number of discrete vortex segments increases to infinity, the behavior of the continuous, bound vortex filament is achieved.

2.1.2 Conditions on the Bound Vortex Filament

It is seen that the integral in Eq. (2.4) contains singularities at $z = z_0$. The limit at that point is

$$\lim_{z \to z_0} d\vec{V}_{\mathrm{LAC}}(z_0) = \frac{-\Gamma(z_0)}{4\pi} \frac{\left[0, f''(z_0), 0\right]}{3\left(1 + f'(z_0)^2\right)^{3/2}(z_0 - z_0)} dz \tag{2.9}$$

Due to the second term in the denominator, this limit is infinite unless $f''(z_0)$ or $\Gamma(z_0)$ is zero, in which case the limit becomes indeterminate. To be useful in lifting-line theory, $\Gamma(z_0)$ must be allowed to remain non-zero. Accordingly, consider the case that

$$f''(z_0) = 0 \tag{2.10}$$

The limit at z_0 can then be found using l'Hospital's rule [25] to be

$$\lim_{z \to z_0} d\vec{V}_{\mathrm{LAC}}(z_0) = \frac{\Gamma(z_0)}{4\pi} \frac{\left[0, f'''(z_0), 0\right]}{6\left(1 + f'(z_0)^2\right)^{3/2}} dz \tag{2.11}$$

which is finite. Therefore, **in order for the total induced velocity at point z_0 to remain finite, $f''(z)$ must be zero in the neighborhood of z_0**. This restricts the cases for which the integral in Eq. (2.4) is guaranteed to be convergent [6, 26, 27].

Recall from Fig. 1.5 that the circulation distribution predicted by Phillips' implementation approaches zero at the root of the wing ($2z/b = 0$). Because Phillips models the locus

of aerodynamic centers using the quarter-chord line of the wing, the concavity is non-zero at the root, and the circulation is accordingly forced to zero, as surmised from Eq. (2.9). Phillips recognized that the true locus of aerodynamic centers of a swept wing deviates from the quarter-chord line and suggested that the accuracy of his method could be improved through a more-accurate modeling of the locus [19]. However, the curved locus suggested in Phillips' paper [19,28] nevertheless violates the condition of zero concavity in Eq. (2.10).

2.1.3 Effective Bound Vortex Shape

Initially, it may appear that the only case in which the condition $f''(z_0) = 0$ is satisfied for every point is that of a linear locus of aerodynamic centers, as in Prandtl's classical implementation. However, by Eq (2.9), it is sufficient that $f''(z_0)$ equal zero only in the neighborhood of z_0, suggesting the possibility of provisionally removing the curve's concavity (i.e. force the second derivative to be zero) at a point, while minimally affecting the original locus of aerodynamic centers at the other spanwise points. Herein, this procedure shall be referred to as *conditional concavity*. The result is an *effective locus of aerodynamic centers* for each point, z_0, along the original locus.

Consider a function, $f(z)$, defining the geometry of the original locus of aerodynamic centers. The line (i.e. curve whose second derivative is zero) tangent to $f(z)$ at the point z_0 is

$$f_{z_0}(z) = f'(z_0)\big(z - z_0\big) + f(z_0) \tag{2.12}$$

Strategically blending $f(z)$ with its tangent line will achieve the goal of conditional concavity. Herein, the blending function $e^{-\sigma(z_0-z)^2}$ will be used. The resulting family of effective loci of aerodynamic centers can be written as

$$\tilde{f}_{z_0}(z) = \big(1 - e^{-\sigma(z_0-z)^2}\big) f(z) + e^{-\sigma(z_0-z)^2}\big(f'(z_0)\big(z - z_0\big) + f(z_0)\big) \tag{2.13}$$

with the first derivative

$$\tilde{f}'_{z_0}(z) = f'(z) + e^{-\sigma(z_0-z)^2}\left(f'(z_0) - f'(z) - 2\sigma(z_0-z)^2\Big(f'(z_0) - \frac{f(z_0)-f(z)}{z_0-z}\Big)\right) \quad (2.14)$$

where σ is a positive, real value. Adjusting σ changes the total influence of the conditional concavity on the original curve, as seen in Fig. 2.2. As the value of σ approaches zero, $\tilde{f}_{z_0}$

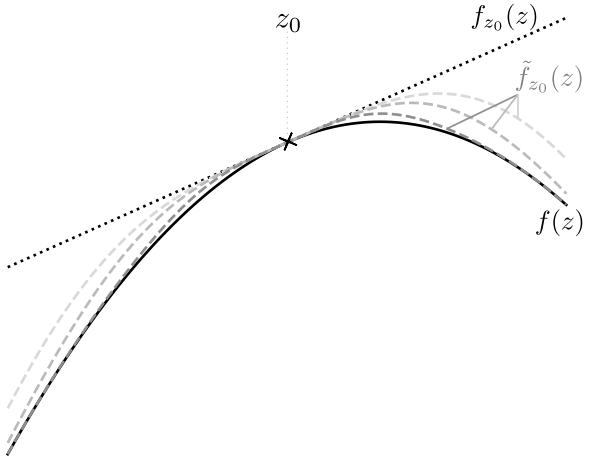

Fig. 2.2: An example application of conditional concavity to a parabola, $f(z)$. Several evaluations of Eq. (2.13), $\tilde{f}_{z_0}(z)$, are shown, each with a different value of σ, dark ($\sigma_{\max}$) to light ($\sigma_{\min}$).

becomes the tangent line f_{z_0}, and as σ tends to infinity, $\tilde{f}_{z_0}$ yields f. The effect of σ is more visualizable if it is written in terms of the distance from the point z_0 at which the contribution of f_{z_0} to $\tilde{f}_{z_0}$ falls below a given threshold. It is convenient to use a threshold of 1.8%, leading to the expression

$$\Delta z = \sqrt{\frac{-\ln(0.018)}{\sigma}} \approx \frac{2}{\sqrt{\sigma}} \quad (2.15)$$

where Δz will herein be referred to as the *blending length*. This distance can be written as the following fraction of the wing's quarter-chord line, to transform the distance along the

z-axis to an approximate, non-dimensional distance along the bound vortex filament

$$\Delta\bar{z} = \frac{\Delta z}{b/\cos\Lambda} \approx \frac{2\cos\Lambda}{\sqrt{b^2\sigma}} \tag{2.16}$$

In summary, to satisfy the condition that the second derivative of the locus of aerodynamic centers be zero at each point, z_0, along the locus of aerodynamic centers, conditional concavity is used to generate a family of effective loci of aerodynamic centers. Each effective locus is designed to satisfy Eq. (2.10) in the neighborhood of z_0, but maintain the geometry of the original curve at the other points along the span. The effectiveness of this method of conditional concavity is explored in Chapter 7.

2.2 Trailing Vortex Sheet

The velocity induced at a point by a single semi-infinite vortex is found by integrating Eq. (2.1) along the ray describing the vortex, resulting in the equation [5]

$$d\vec{V} = \frac{\Gamma}{4\pi}\frac{\vec{u}_\infty \times \vec{r}}{r\left(r - \vec{u}_\infty \cdot \vec{r}\right)} \tag{2.17}$$

where r is the magnitude of $\vec{r}$, and $\vec{u}_\infty$ is the unit vector defining the direction of the ray. The semi-infinite vortices included in the trailing vortex sheet are free vortices aligned with the freestream, so $\vec{u}_\infty$ is therefore the unit vector in the direction of the freestream velocity. The velocity induced at a point, z_0, along the locus of aerodynamic centers by the entire trailing vortex sheet is then found by integrating Eq. (2.17) along the span of the wing, i.e.

$$\vec{V}_{\mathrm{TV}}(z_0) = \int_{-b/2}^{b/2} \frac{\Gamma'(z)}{4\pi} \frac{\left[u_y,\ u_z\frac{f(z_0)-f(z)}{z_0-z} - u_x,\ -u_y\frac{f(z_0)-f(z)}{z_0-z}\right]}{\left(1+\left(\frac{f(z_0)-f(z)}{z_0-z}\right)^2\right)\left(1-\frac{u_z+u_x\frac{f(z_0)-f(z)}{z_0-z}}{\sqrt{1+\left(\frac{f(z_0)-f(z)}{z_0-z}\right)^2}}\right)(z_0-z)}dz \tag{2.18}$$

Recall that the strength of the semi-infinite vortex sheet is equal to the change in the circulation distribution of the bound vortex filament (i.e. $\Gamma'(z)$).

2.2.1 Numerical Implementation of Semi-Infinite Trailing Vortices

Now, consider the influence of a finite number of semi-infinite vortices, such as those used in Phillips' implementation to approximate the trailing vortex sheet. Using the notation in Fig. 2.3, the velocity induced at a point by a single semi-infinite vortex can be

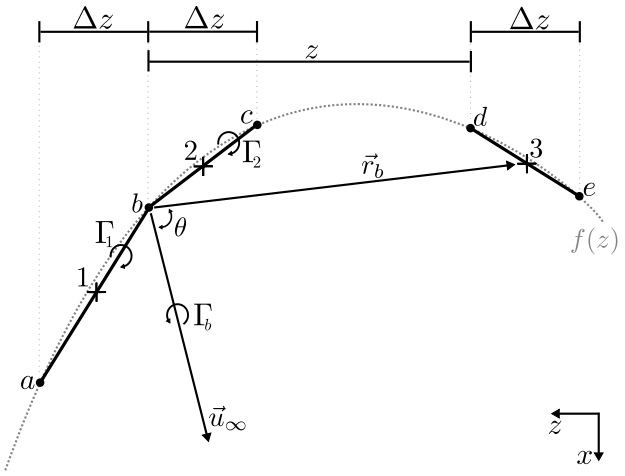

Fig. 2.3: The geometry used to define the velocity induced at a point by a single semi-infinite vortex. ($\Delta z > 0$ and $z = z_b - z_d$)

described as

$$d\vec{V}_3 = \frac{\Gamma_b}{4\pi} \frac{\vec{u}_\infty \times \vec{r}_b}{r_b\big(r_b - \vec{u}_\infty \cdot \vec{r}_b\big)} \tag{2.19}$$

where r_b is the magnitude of $\vec{r}_b$. Using point b as a reference, the vectors $\vec{u}_\infty$ and $\vec{r}_b$ are defined as

$$\vec{u}_\infty = \Big[u_x, u_y, u_z\Big] \tag{2.20}$$

$$\vec{r}_b = \Big[\frac{f(z_d) + f(z_d - \Delta z)}{2} - f(z_b), 0\,, -(z_b - z_d) - \frac{\Delta z}{2}\Big] \tag{2.21}$$

and Γ_b is defined as

$$\Gamma_b = \Gamma_1 - \Gamma_2 = \Gamma\Big(z_b + \frac{\Delta z}{2}\Big) - \Gamma\Big(z_b - \frac{\Delta z}{2}\Big) \tag{2.22}$$

As $\Delta z \to 0$, Eq. (2.19) becomes

$$\lim_{\Delta z \to 0} \vec{v}_3 = \frac{\Gamma'(z_b)}{4\pi} \frac{\left[u_y,\, u_z \frac{f(z_d)-f(z_b)}{z_d-z_b} - u_x,\, -u_y \frac{f(z_d)-f(z_b)}{z_d-z_b}\right] dz}{\sqrt{1+\left(\frac{f(z_d)-f(z_b)}{z_d-z_b}\right)^2}\left(\sqrt{1+\left(\frac{f(z_d)-f(z_b)}{z_d-z_b}\right)^2} - u_z - u_x \frac{f(z_d)-f(z_b)}{z_d-z_b}\right)(z_d - z_b)} \tag{2.23}$$

which matches Eq. (2.18), signifying that the discrete number of semi-infinite vortices replicate the influence of a vortex sheet as the number of discrete semi-infinite vortices approaches infinity.

2.2.2 Conditions on the Trailing Vortex Sheet

As was the case with the integral describing the influence of the vortex filament, given in Eq. (2.4), Eq. (2.18) also contains singularities. For example, at $z = z_0$ the limit of Eq. (2.17) is

$$\lim_{z \to z_0} d\vec{V}_{\mathrm{TV}}(z_0) = \frac{\Gamma'(z_0)}{4\pi} \frac{\left[u_y,\, u_z f'(z_0) - u_x,\, -u_y f'(z_0)\right]}{\left(1+f'(z_0)^2\right)\left(1 - \frac{u_z + u_x f'(z_0)}{\sqrt{1+f'(z_0)^2}}\right)(z_0 - z_0)} dz \tag{2.24}$$

which is infinite. A second singularity occurs when a portion of the trailing vortex sheet lays directly on the point at which the induced velocity is to be calculated. In that case, the second term in the denominator of Eq. (2.18) becomes zero, and the limit is again infinite. Because of the existence of singularities in the integral, neither the convergence of the integral nor the existence of its principle value can be guaranteed for an arbitrary case [6, 26, 27].

To identify conditions for which the integral in Eq. (2.18) can be evaluated, assume that the influence of the portion of the trailing vortex sheet in the vicinity of z_0 can be adequately approximated as the influence of a semi-infinite vortex sheet of constant strength per unit length, γ_s, described by Hunsaker and Phillips [29] and shown in Fig. 2.4.

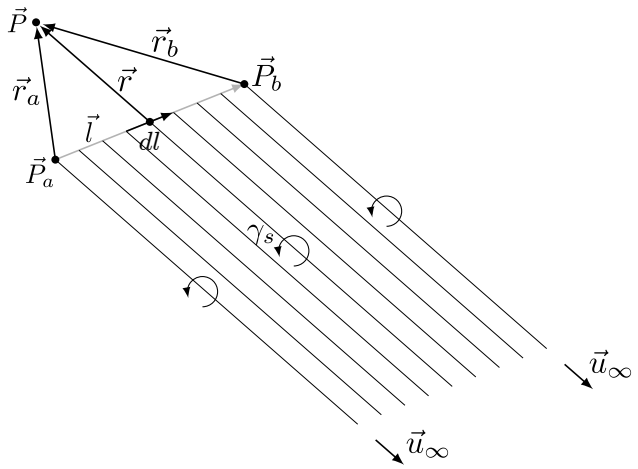

Fig. 2.4: The geometry used to define the velocity induced at a point, $\vec{P}$, by a semi-infinite vortex sheet of constant strength per unit length, γ_s.

In the case that the point at which the influence is being calculated, $\vec{P}$, is located on the line

$$\vec{l} = \vec{P}_b - \vec{P}_a \tag{2.25}$$

such that

$$\vec{P} = \vec{P}_a + \xi\vec{l} \tag{2.26}$$

where ξ is a dimensionless distance along $\vec{l}$, the vector $\vec{r}$ is defined as

$$\vec{r} = (\xi - \zeta)\vec{l} \tag{2.27}$$

with magnitude

$$r = |\xi - \zeta|l \tag{2.28}$$

where ζ is a dimensionless distance along $\vec{l}$ from $\vec{P}_a$ to $\vec{P}_b$, such that $0 < \zeta < 1$. The influence on such a point can be described by the integral

$$\begin{aligned}\vec{V}_s &= \frac{\gamma_s l}{4\pi}\int_0^1 \frac{(\xi-\zeta)(\vec{u}_\infty \times \vec{l})}{|\xi-\zeta| l\Big(|\xi-\zeta| l - (\xi-\zeta)(\vec{u}_\infty \cdot \vec{l})\Big)} d\zeta \\ &= \frac{\gamma_s l}{4\pi}\int_0^1 \frac{\text{sign}(\xi-\zeta)(\vec{u}_\infty \times \vec{l})}{l\Big(l - \text{sign}(\xi-\zeta)(\vec{u}_\infty \cdot \vec{l})\Big)|\xi-\zeta|} d\zeta \end{aligned} \tag{2.29}$$

which can be separated into the cases

$$\vec{V}_s = \frac{\gamma_s l}{4\pi}\begin{cases} \int_0^1 \frac{-(\vec{u}_\infty \times \vec{l})}{l\Big(l+(\vec{u}_\infty \cdot \vec{l})\Big)(-\xi+\zeta)} d\zeta & \xi < 0 \\ \lim_{\epsilon \to 0^+}\left[\int_0^{\xi-\epsilon} \frac{(\vec{u}_\infty \times \vec{l})}{l\Big(l-(\vec{u}_\infty \cdot \vec{l})\Big)(\xi-\zeta)} d\zeta + \int_{\xi+\epsilon}^1 \frac{-(\vec{u}_\infty \times \vec{l})}{l\Big(l+(\vec{u}_\infty \cdot \vec{l})\Big)(-\xi+\zeta)} d\zeta\right] & 0 < \xi < 1 \\ \int_0^1 \frac{(\vec{u}_\infty \times \vec{l})}{l\Big(l-(\vec{u}_\infty \cdot \vec{l})\Big)(\xi-\zeta)} d\zeta & 1 < \xi \end{cases} \tag{2.30}$$

Evaluating Eq. (2.30), the total influence of the sheet can be written as

$$\vec{V}_s = \frac{\gamma_s l}{4\pi}\begin{cases} \frac{(\vec{u}_\infty \times \vec{l})}{l\Big(l+(\vec{u}_\infty \cdot \vec{l})\Big)} \ln\left(\frac{\xi}{\xi-1}\right) & \xi < 0 \\ \begin{cases} \frac{(\vec{u}_\infty \times \vec{l})}{l^2} \ln\left(\frac{\xi}{1-\xi}\right) & \text{if } \vec{u}_\infty \cdot \vec{r} = 0 \\ \text{undefined} & \text{if } \vec{u}_\infty \cdot \vec{r} \neq 0 \end{cases} & 0 < \xi < 1 \\ \frac{(\vec{u}_\infty \times \vec{l})}{l\Big(l-(\vec{u}_\infty \cdot \vec{l})\Big)} \ln\left(\frac{\xi}{\xi-1}\right) & 1 < \xi \end{cases} \tag{2.31}$$

Notice that, in the case that $0 < \xi < 1$, **it is only when**

$$\boldsymbol{\vec{u}_\infty \cdot \vec{r} = 0} \tag{2.32}$$

that the principle value of the integral exists. Prandtl's classical implementation of lifting-line theory satisfies Eq. (2.32) because it models a linear locus of aerodynamic

centers in freestream conditions without sideslip. Thus, the line representing the locus of aerodynamic centers is at all points perpendicular to the trailing vortex sheet.

2.2.3 Jointed Trailing Vortex Sheet

To allow for a more-general locus of aerodynamic center shape and freestream conditions, the classic trailing vortex sheet used in Prandtl's lifting-line implementation must be modified. One simple modification is made by *jointing* each trailing vortex, such that there is a finite segment of the trailing vortex perpendicular to, and in the same plane as, $f(z)$, and a semi-infinite portion aligned with the freestream. It is possible that other trailing vortex geometries would also achieve the desired effect[1], but this simple joint maintains the simplicity of lifting-line theory.

Consider a vortex sheet, similar to the one shown in Fig. 2.4, but being comprised of finite vortex segments instead of semi-infinite vortices, shown in Fig. 2.5. The finite vortex

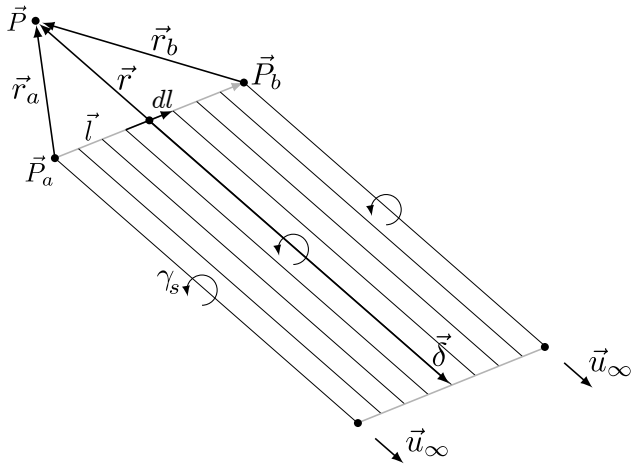

Fig. 2.5: The geometry used to define the velocity induced at a point, $\vec{P}$, by a finite vortex sheet of constant strength, γ_s.

sheet is defined by the vector

$$\vec{\delta} = \delta \vec{u}_\infty \tag{2.33}$$

And, as before, $\vec{P}$, $\vec{l}$, ξ, $\vec{r}$, and ζ are defined by Eqs. (2.25)–(2.28). The condition of interest

[1] A detailed description of the influence of a parabolic vortex segment is found in Chapter 3

is when $\vec{u}_\infty \cdot \vec{r} = 0$. In that case, the influence on a point along $\vec{l}$ can be described by the integral

$$\begin{aligned}\vec{V}_s &= \frac{\gamma_s l}{4\pi}\int_0^1 \frac{(r+|\vec{r}-\vec{\delta}|)(\vec{r}\times(\vec{r}-\vec{\delta}))}{r|\vec{r}-\vec{\delta}|\Big(r|\vec{r}-\vec{\delta}|+\vec{r}\cdot(\vec{r}-\vec{\delta})\Big)}d\zeta \\ &= \frac{\gamma_s l}{4\pi}\int_0^1 \frac{\delta(\xi-\zeta)}{|\xi-\zeta|^2 l\sqrt{|\xi-\zeta|^2 l^2+\delta^2}}d\zeta \\ &= \frac{\gamma_s \delta}{4\pi}\int_0^1 \frac{\text{sign}(\xi-\zeta)}{|\xi-\zeta|\sqrt{|\xi-\zeta|^2 l^2+\delta^2}}d\zeta \end{aligned} \tag{2.34}$$

which can be separated into the cases

$$\vec{V}_s = \frac{\gamma_s \delta}{4\pi}\begin{cases} \int_0^1 \frac{-1}{(-\xi+\zeta)\sqrt{(-\xi+\zeta)^2 l^2+\delta^2}}d\zeta & \xi<0 \\ \lim_{\epsilon\to 0^+}\left[\int_0^{\xi-\epsilon}\frac{1}{(\xi-\zeta)\sqrt{(\xi-\zeta)^2 l^2+\delta^2}}d\zeta + \int_{\xi+\epsilon}^1 \frac{-1}{(-\xi+\zeta)\sqrt{(-\xi+\zeta)^2 l^2+\delta^2}}d\zeta\right] & 0<\xi<1 \\ \int_0^1 \frac{1}{(\xi-\zeta)\sqrt{(\xi-\zeta)^2 l^2+\delta^2}}d\zeta & 1<\xi \end{cases} \tag{2.35}$$

Evaluating Eq. (2.35), the total influence of the sheet can be written as

$$\vec{V}_s = \frac{\gamma_s \delta}{4\pi}\begin{cases} \frac{1}{\delta}\ln\left(\frac{\xi\left(\delta+\sqrt{(\xi-1)^2 l^2+\delta^2}\right)}{(\xi-1)\left(\delta+\sqrt{\xi^2 l^2+\delta^2}\right)}\right) & \xi<0 \\ \frac{1}{\delta}\ln\left(\frac{\xi\left(\delta+\sqrt{(\xi-1)^2 l^2+\delta^2}\right)}{(1-\xi)\left(\delta+\sqrt{\xi^2 l^2+\delta^2}\right)}\right) & 0<\xi<1 \\ \frac{1}{\delta}\ln\left(\frac{\xi\left(\delta+\sqrt{(\xi-1)^2 l^2+\delta^2}\right)}{(\xi-1)\left(\delta+\sqrt{\xi^2 l^2+\delta^2}\right)}\right) & 1<\xi \end{cases} \tag{2.36}$$

As with the semi-infinite vortex sheet, if $\vec{u}_\infty \cdot \vec{r} = 0$, the influence of a sheet of finite vortex segments at a point on $\vec{l}$ is defined for all values of ξ. Therefore, it is concluded that the use of a finite vortex sheet perpendicular to the locus of aerodynamic centers will result in a finite induced velocity and will be suitable for use in the general implementation of lifting-line theory.

More generally, using the notation in Fig. 2.6, the velocity induced at a point by the finite segment of the jointed vortex can be described by integrating Eq. (2.1) along $\vec{r}_\delta$, to

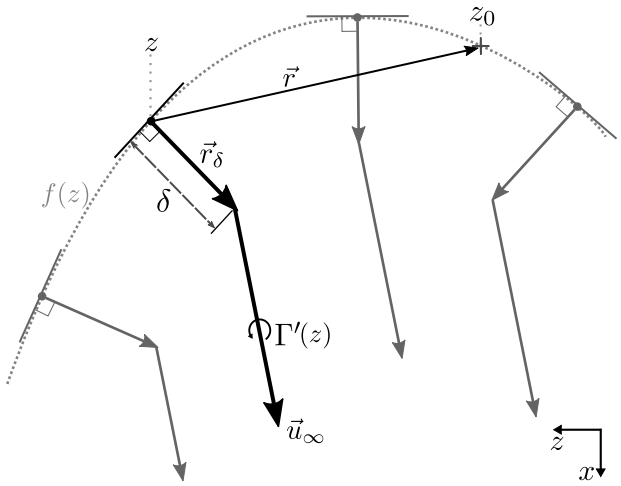

Fig. 2.6: The geometry used to define the velocity induced at a point, z_0 by a single semi-infinite, jointed vortex.

give

$$d\vec{V}_\delta(z_0) = \frac{\Gamma'(z)}{4\pi} \frac{(r + |\vec{r} - \vec{r}_\delta|)\big(\vec{r} \times (\vec{r} - \vec{r}_\delta)\big)}{r|\vec{r} - \vec{r}_\delta|\big(r|\vec{r} - \vec{r}_\delta| + \vec{r} \cdot (\vec{r} - \vec{r}_\delta)\big)} \tag{2.37}$$

where

$$\vec{r} = \Big[f(z_0) - f(z), 0, z_0 - z\Big] \tag{2.38}$$

$$\vec{r}_\delta = \Big[1, 0, -f'(z)\Big] \frac{\delta}{\sqrt{1 + f'(z)^2}} \tag{2.39}$$

and δ is the length of the finite vortex segment. Similarly, the velocity induced at a point by the semi-infinite portion of the trailing vortex can be described by Eq. (2.17) as

$$d\vec{V}_{\mathrm{TV}'}(z_0) = \frac{\Gamma'(z)}{4\pi} \frac{\vec{u}_\infty \times (\vec{r} - \vec{r}_\delta)}{|\vec{r} - \vec{r}_\delta|\big(|\vec{r} - \vec{r}_\delta| - \vec{u}_\infty \cdot (\vec{r} - \vec{r}_\delta)\big)} \tag{2.40}$$

Thus, the total velocity induced at a spanwise location, z_0, is found by integrating the influence of the sheet of finite segments and the influence of the sheet of semi-infinite

vortices along the span

$$\begin{aligned}\vec{V}_{\mathrm{TV}_\delta}(z_0) &= \int_{-b/2}^{b/2} \Big(d\vec{V}_\delta(z_0) + d\vec{V}_{\mathrm{TV}'}(z_0) \Big) dz \\ &= \int_{-b/2}^{b/2} \frac{\Gamma'(z)}{4\pi} \Bigg(\frac{(r + |\vec{r} - \vec{r}_\delta|)\big(\vec{r} \times (\vec{r} - \vec{r}_\delta)\big)}{r|\vec{r} - \vec{r}_\delta|\big(r|\vec{r} - \vec{r}_\delta| + \vec{r} \cdot (\vec{r} - \vec{r}_\delta)\big)} \\ &\qquad + \frac{\vec{u}_\infty \times (\vec{r} - \vec{r}_\delta)}{|\vec{r} - \vec{r}_\delta|\big(|\vec{r} - \vec{r}_\delta| - \vec{u}_\infty \cdot (\vec{r} - \vec{r}_\delta)\big)} \Bigg) dz \end{aligned} \tag{2.41}$$

Notice that, as $z \to z_0$, only the first term in the integral is indeterminate because r approaches zero. The observations made of the semi-infinite vortex sheet of constant strength, in Eq. (2.31), and the sheet of constant-strength vortex segments, in Eq. (2.36), can thus be extended to hypothesize that, in order for the total induced velocity at point z_0 to remain finite, the condition in Eq. (2.32), i.e.

$$\vec{r}_\delta(z_0) \perp \vec{r}(z) \tag{2.42}$$

must exist in the neighborhood of z_0. The validity of this conjecture is demonstrated in Chapter 7.

2.3 Total Induced Velocity

Summing the integrals that describe the velocity induced by the trailing vortex sheet, given in Eq (2.41), and the bound vortex filament, given in Eq (2.4), results in the total velocity induced at the spanwise location z_0

$$\vec{V}_i(z_0) = \int_{-b/2}^{b/2} \Big(d\vec{V}_{\mathrm{LAC}}(z_0) + d\vec{V}_{\mathrm{TV}_\delta}(z_0) \Big) dz \tag{2.43}$$

As discussed in the previous sections, this integral contains singularities that can cause the integral to diverge. However, convergence of the integral can be assured if, in the neighborhood of z_0, the conditions in Eq. 2.10, i.e.

$$f''(z_0) = 0$$

and Eq. 2.42, i.e.

$$\vec{r}_\delta(z_0) \perp \vec{r}(z)$$

are met. The model for induced velocity given in Eq. (2.43) can therefore by used to develop the general implementation of lifting-line theory shown in Chapter 6.

CHAPTER 3

PARABOLIC VORTEX SEGMENT

In Chapter 2, the influence of linear vortices is discussed—a finite vortex segment and a semi-infinite vortex. Many potential-flow based methods—including Prandtl's, Phillips', and Weissinger's implementations of lifting-line theory—use these and other types of linear potential-flow elements to approximate non-linear strength distributions or geometries [1, 2, 13–17, 19, 29–32]. Applying higher-order, non-linear potential-flow elements in these algorithms can achieve higher accuracy for a fewer number of elements, potentially reducing the computational cost of the method [10, 11, 33–39].

Work to predict the influence of various curved vortex segments has been done before. Yoon and Heister developed analytic predictions for the influence of a thin vortex ring [38]. Beyer et al. developed a closed-form prediction for the influence of a circular-arc vortex segment with a cut-off radius [10]. Bliss et al. also predicted the self-influence of a circular-arc vortex segment, and predicted the influence of a symmetric parabolic vortex segment using simplifying approximations [11]. The work performed in this chapter results in a closed-form prediction of the influence of a non-symmetric parabolic vortex segment in three-dimensions, allowing for more applicability than circular-arc segments, without the approximations made by Bliss et al. The computational cost of the resulting closed-form prediction is evaluated against its accuracy to determine its benefit to the general implementation of lifting-line theory.

3.1 Definition of a Parabolic Vortex Segment

As discussed in Chapter 2, the Biot-Savart law describes the differential velocity, $d\vec{V}$, induced by the differential element, $d\vec{l}$, of a vortex filament in three dimensions [5, 24]

$$d\vec{V} = \frac{\Gamma}{4\pi} \frac{d\vec{l} \times \vec{r}}{|\vec{r}|^3} \tag{3.1}$$

where Γ is the strength of the vortex, and $\vec{r}$ is the position vector from the differential vortex segment to the point, $\vec{x}$, at which the differential induced velocity is calculated. If the vortex follows the parameterized curve, $\vec{f}(t)$, then Eq. (3.1) can be rewritten as

$$d\vec{V} = \frac{\Gamma}{4\pi}\frac{d\vec{f}(t) \times \big(\vec{x} - \vec{f}(t)\big)}{|\vec{x} - \vec{f}(t)|^3} \tag{3.2}$$

where $d\vec{f}(t)$ is the derivative of $\vec{f}(t)$ with respect to the parameter t.

Consider a parabolic vortex segment beginning at point $\vec{f_0}$ and ending at point $\vec{f_1}$, as shown in Fig. 3.1, and let it be defined by the parameterized curve

$$\begin{aligned} \vec{f}(t) &= (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t^2 + \vec{f}_0' t + \vec{f}_0 \\ d\vec{f}(t) &= \big(2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t + \vec{f}_0'\big)dt \end{aligned} \tag{3.3}$$

for $0 \leq t \leq 1$. Using Eq. (3.3) in Eq. (3.2), results in the expression

$$d\vec{V} = \frac{\Gamma}{4\pi}\frac{\big(2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t + \vec{f}_0'\big)dt \times \big(\vec{x} - (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t^2 - \vec{f}_0' t - \vec{f}_0\big)}{|\vec{x} - (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t^2 - \vec{f}_0' t - \vec{f}_0|^3} \tag{3.4}$$

which can be rewritten using the definitions shown in Fig. 3.1 as

$$d\vec{V} = \frac{\Gamma}{4\pi}\frac{\big(\vec{f}_0' \times (\vec{r}_0 - \vec{r}_1)\big)t^2 - 2\big((\vec{r}_1 + \vec{f}_0') \times \vec{r}_0\big)t + \vec{f}_0' \times \vec{r}_0}{\left(\begin{array}{l}(\vec{r}_0 - \vec{r}_1 - \vec{f}_0')^2 t^4 + 2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0') \cdot \vec{f}_0' t^3 \\ \quad + \big(f_0'^2 - 2\vec{r}_0 \cdot (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')\big)t^2 - 2\vec{r}_0 \cdot \vec{f}_0' t + r_0^2\end{array}\right)^{3/2}} \tag{3.5}$$

where $r_0 = |\vec{r}_0|$, $r_1 = |\vec{r}_1|$, and $f_0' = |\vec{f}_0'|$. The total velocity induced at $\vec{x}$ by the vortex segment is calculated as Eq. (3.5) is integrated along the length of the vortex segment, resulting in an integral of the form

$$\vec{V} = \frac{\Gamma}{4\pi}\int_0^1 \frac{\Big(At^2 + Bt + C\Big)dt}{\Big(Dt^4 + Et^3 + Ft^2 + Gt + H\Big)^{3/2}} \tag{3.6}$$

where

$$A = \vec{f}_0' \times (\vec{r}_0 - \vec{r}_1), \quad B = -2(\vec{r}_1 + \vec{f}_0') \times \vec{r}_0, \quad C = \vec{f}_0' \times \vec{r}_0,$$

$$D = (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')^2, \quad E = 2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0') \cdot \vec{f}_0', \quad F = f_0'^2 - 2\vec{r}_0 \cdot (\vec{r}_0 - \vec{r}_1 - \vec{f}_0'),$$

$$G = -2\vec{r}_0 \cdot \vec{f}_0', \quad H = r_0^2$$

Eq. (3.6) belongs to the family of elliptical integrals, for which special integration techniques are required to obtain a general, closed-form solution.

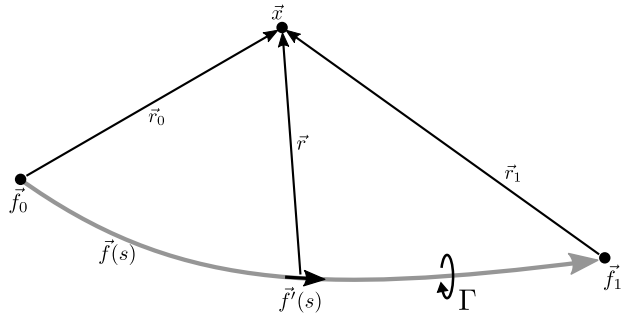

Fig. 3.1: The geometry used to define velocity induced by a parabolic vortex segment at the point $\vec{x}$.

3.1.1 Previous Work

Before the solution to the general, parabolic vortex segment is discussed, it is important to consider the work previously completed in the prediction of vortex segment influence. This previous work provides a means against which to verify the fidelity of mathematical derivations and the accuracy of numerical results. First, the simplifying case of a linear vortex segment is described. Then, the work of Bliss et al. is summarized. Substantial work in the field of circular arc vortex segments has been performed [10, 11, 33–38], but will not be described herein because it does not provide a direct means of validation for this work.

3.1.1.1 Linear Vortex Segment

In the special case that $\vec{f}_0' = \vec{r}_0 - \vec{r}_1$, the parabolic vortex segment becomes a line that extends from the point $\vec{f}_0$ to the point $\vec{f}_1$. Thus, for this linear vortex segment,

$$\begin{aligned} \vec{f}(t) &= \vec{f}_0' t + \vec{f}_0 \\ d\vec{f}(t) &= \vec{f}_0' dt \end{aligned} \tag{3.7}$$

for $0 \leq t \leq 1$. Using Eq. (3.7) en lieu of Eq. (3.3), Eq. (3.6) becomes

$$\vec{V} = \frac{\Gamma}{4\pi} \int_0^1 \frac{\big(\vec{r}_0 \times \vec{r}_1\big) dt}{\Big(\big(r_0^2 + r_1^2 - 2(\vec{r}_0 \cdot \vec{r}_1)\big)t^2 - 2\big(r_0^2 - \vec{r}_0 \cdot \vec{r}_1\big)t + r_0^2\Big)^{3/2}} \tag{3.8}$$

which can be evaluated to yield

$$\vec{V} = \frac{\Gamma\left(\vec{r}_0 \times \vec{r}_1\right)}{4\pi} \left(\frac{r_0 + r_1}{\big(\vec{r}_0 \cdot \vec{r}_1 + r_0 r_1\big) r_0 r_1} \right) \tag{3.9}$$

The simplicity of the closed-form solution described in Eq. (3.9) allows it to readily be used, as was seen in Chapter 2. Eq. (3.9) also provides a check against which the general solution for the influence of a parabolic vortex segment obtained in this chapter can be compared. A correct solution to Eq. (3.6) will return Eq. (3.9) in the case that $\vec{f}_0' = \vec{r}_0 - \vec{r}_1$.

3.1.1.2 Previous Approximation of Parabolic Vortex Segment

The influence of a parabolic vortex segment has been approximated by Bliss et al., for the parabolic arc described by the equation [11]

$$y = \varepsilon x^2 \tag{3.10}$$

where ε is defined in terms of the radius of the circular arc, R_0, and the arc's central angle, θ_0,

$$\varepsilon = \frac{\cos\frac{\theta_0}{2} - 1}{R_0 \sin^2\frac{\theta_0}{2}} \tag{3.11}$$

and x varies from $-\ell$ to ℓ. With this definition for the arc, the differential arc-length vector can be written

$$d\vec{l} = \big[1, 2\varepsilon x, 0\big]\, dx \tag{3.12}$$

and the distance from a point on the arc to the point at which the influence is calculated, $\big[x_0, y_0, z_0\big]$, is

$$\vec{r} = \big[x_0 - x, y_0 - \varepsilon x^2, z_0\big] \tag{3.13}$$

Bliss et al. then describe the influence of the arc at a point, using the Biot-Savart law, as [11]

$$\begin{aligned}\vec{V} &= \frac{\Gamma}{4\pi}\int_C \frac{d\vec{l}\times\vec{r}}{|\vec{r}|^3} \\ &= \frac{\Gamma}{4\pi}\int_{-\ell}^{\ell} \frac{\big[2\varepsilon z_0 x,\ -z_0,\ -2\varepsilon x x_0 + \varepsilon x^2 + y_0\big]}{\Big(\varepsilon^2 x^4 + (1-2\varepsilon y_0)x^2 + (-2x_0)x + (x_0^2 + y_0^2 + z_0^2)\Big)^{3/2}} dx\end{aligned} \tag{3.14}$$

In order to evaluate Eq. (3.14), Bliss et al. model the quartic term, $\varepsilon^2 x^4$, by the quadratic

$$\varepsilon^2 x^4 \approx \varepsilon^2\big(F_2(x_0)x^2 + F_1(x_0)x + F_0(x_0)\big) \tag{3.15}$$

where F_2, F_1, and F_0 are functions of x_0, resulting in the equation

$$\vec{V} = \frac{\Gamma}{4\pi}\int_{-\ell}^{\ell} \frac{\big[2\varepsilon z_0 x,\ -z_0,\ -2\varepsilon x x_0 + \varepsilon x^2 + y_0\big]}{\Big((1-2\varepsilon y_0 + \varepsilon^2 F_2)x^2 + (-2x_0 + \varepsilon^2 F_1)x + (x_0^2 + y_0^2 + z_0^2 + \varepsilon^2 F_0)\Big)^{3/2}} dx \tag{3.16}$$

Evaluating this simpler integral yields an expression in which F_2, F_1, and F_0 that must be tuned, depending on x_0 and $|\vec{r}|$, to accurately replicate the original integral. Bliss et al. describe their tuning methodology in detail, and it will not be described herein [11].

Equation (3.10) specifies a vortex segment that is equivalent to the special case of Eq. (3.3), in which

$$\vec{f}_0 = \big[-\ell, \varepsilon\ell^2, 0\big] \tag{3.17}$$

$$\vec{r}_0 - \vec{r}_1 = \big[2\ell, 0, 0\big] \tag{3.18}$$

$$\vec{f}_0' = \left[2\ell, -4\varepsilon\ell^2, 0\right] \tag{3.19}$$

Therefore, for this case, the general solution for the influence of a parabolic vortex segment may be compared against the results of Eq. (3.16), as well as a numerical evaluation of Eq. (3.6). Such a comparison will provide insight into the accuracy of the general solution and the approximate solution made by Bliss et al.

3.2 Evaluation of the Induced Velocity using Elliptic Integrals

The evaluation of an integral of the form

$$\int \frac{At^2 + Bt + C}{\left(Dt^4 + Et^3 + Ft^2 + Gt + H\right)^{3/2}} dt \tag{3.20}$$

requires techniques beyond those used in traditional integration[1]. The original integral is rewritten in terms of symmetric integrals, which are then redefined based on the relation between genus-one curves and their Jacobian varieties, and finally related to hypergeometric functions. Note that, for generality, the constant multiplier in Eq. (3.6), $\Gamma/4\pi$, has been factored out of Eq. (3.20) and does not appear throughout this derivation. It is re-included for the discussion of this work's application and results.

3.2.1 Reduction to a Standard Symmetric Integral

Consider, first, a genus-one curve, $\mathcal{C}$, given as a quartic curve with affine coordinates $(t, s) \in \mathbb{C}^2$ by

$$\mathcal{C}: \quad s^2 = \prod_{i=1}^{4} \left(\alpha_i + \beta_i t\right) = Dt^4 + Et^3 + Ft^2 + Gt + H \tag{3.21}$$

where it is always assumed that s^2 only has simple zeros in t. Let $R(s, t)$ be a rational function of s and t, containing at least one odd power of s. Expressions of the form

$$\int R(s, t)\, dt \tag{3.22}$$

[1]The mathematical process described in Section 3.2 is the result of work performed by committee member Andreas Malmendier, Ph.D. and is used with permission.

are called *elliptic integrals*. Because s^2 is a polynomial in t, one can carry out a partial fraction decomposition and write

$$R(s,t) = \frac{\rho}{s} + \sigma \tag{3.23}$$

where ρ and σ are functions of t alone. Herein, the case where $\sigma = 0$ is of interest. In fact, the elliptic integral in Eq. (3.20), is of the general form

$$\mathcal{F} = \int_Y^X \frac{p_2(t)}{s^2}\frac{dt}{s} = \int_Y^X \frac{At^2 + Bt + C}{\Big(Dt^4 + Et^3 + Ft^2 + Gt + H\Big)^{3/2}}dt \tag{3.24}$$

Assume that the limits of integration are real (i.e. $X, Y \in \mathbb{R}$) and that for $1 \leq i \leq 4$ the line segments with endpoints $\alpha_i + \beta_i X$ and $\alpha_i + \beta_i Y$ lie entirely within the complex plane cut along the negative real axis.

The transformation, Ψ, between the variables (u, v) and (t, s), given by

$$\Psi: \quad z_i = \frac{\alpha_i + \beta_i Y}{\alpha_i + \beta_i X}\,, \quad u = -\frac{t - Y}{t - X}\,, \quad v = \frac{(X - Y)^2 s}{s_0 (X - t)^2} \tag{3.25}$$

where $s_0 := s(X) = \sqrt{(\alpha_1 + \beta_1 X)\cdots(\alpha_4 + \beta_4 X)}$, transforms the genus one curve $\mathcal{C}$ into the normalized quartic equation with affine coordinates $(u, v) \in \mathbb{C}^2$, given by

$$\mathcal{C}: \quad v^2 = \big(z_1 + u\big)\big(z_2 + u\big)\big(z_3 + u\big)\big(z_4 + u\big) \tag{3.26}$$

It is easy to check that the transformation described in Eq. (3.25) relates the holomorphic differentials (via pull back),

$$\frac{dt}{s} = \Psi^* \left(\frac{X - Y}{s_0}\frac{du}{v} \right) \tag{3.27}$$

as well as meromorphic differentials,

$$\frac{p_2(t)}{s^2}\frac{dt}{s} = \Psi^* \left(\frac{X - Y}{s_0^3}\frac{du}{v}\frac{(1 + u)^2\big(A'(1 + u)^2 + B'(1 + u) + C'\big)}{v^2} \right) \tag{3.28}$$

where

$$\begin{aligned} A' &= p_2(X) = AX^2 + BX + C\,, \\ B' &= -p_2'(X)(X-Y) = -(2AX+B)(X-Y)\,, \\ C' &= \frac{1}{2}p_2''(X)(X-Y)^2 = A(X-Y)^2 \end{aligned} \tag{3.29}$$

Thus, the integral $\mathcal{F}$ is rewritten in terms of the new variables u and v, related by Eq. (3.26), yielding

$$\mathcal{F} = \int_Y^X \frac{p_2(t)}{s^2}\frac{dt}{s} = \frac{X-Y}{s_0^3}\int_0^\infty \frac{(1+u)^2\big(A'(1+u)^2 + B'(1+u) + C'\big)}{v^2}\frac{du}{v} \tag{3.30}$$

The parameters z_i in Eq. (3.26) are assumed to satisfy $\arg(z_i) < \pi$ for $1 \le i \le 4$, so that the integrals on both sides of Eq. (3.30) are well defined. The integral in Eq. (3.30) can be further decomposed as

$$\mathcal{F} = \frac{X-Y}{s_0^3}\left(\mu_0\int_0^\infty \frac{du}{v} - \sum_{i=1}^4 \mu_i \frac{(z_i-1)^2}{\prod_{i\neq j}(z_i-z_j)}\int_0^\infty \frac{1}{z_i+u}\frac{du}{v}\right) \tag{3.31}$$

where

$$\mu_i = \begin{cases} A' & \text{for } i = 0 \\ A'z_i^2 - (2A'+B')z_i + (A'+B'+C') & \text{for } 1 \le i \le 4 \end{cases} \tag{3.32}$$

Consider the *symmetric integral* defined by Gradshteyn and Ryzhik as [40]

$$R_{\mathcal{F}} = \frac{1}{2}\int_0^\infty \frac{du}{\sqrt{(z_1+u)(z_2+u)(z_3+u)(z_4+u)}} \tag{3.33}$$

The derivatives of the symmetric integral with respect to parameters z_i are given by

$$\frac{\partial}{\partial z_i}R_{\mathcal{F}}(z_1,z_2,z_3,z_4) = -\frac{1}{4}\int_0^\infty \frac{\partial_{z_i}(v^2)}{v^2}\frac{du}{v} = -\frac{1}{4}\int_0^\infty \frac{1}{z_i+u}\frac{du}{v} \tag{3.34}$$

Therefore, the computation of the integral in Eq. (3.31) is reduced to the computation of a standard symmetric integral and its partial derivatives

$$\mathcal{F} = \frac{2(X-Y)}{s_0^3}\left(\mu_0 + \sum_{i=1}^{4} 2\mu_i \frac{(z_i-1)^2}{\prod_{i\neq j}(z_i - z_j)}\frac{\partial}{\partial z_i}\right) R_{\mathcal{F}} \tag{3.35}$$

Moreover, without loss of any generality, it is assumed that $X = 1$ and $Y = 0$.

Thus, in summary, to evaluate the elliptic integral of the form

$$\mathcal{F} = \int_0^1 \frac{At^2 + Bt + C}{\left(Dt^4 + Et^3 + Ft^2 + Gt + H\right)^{3/2}} dt \tag{3.36}$$

it is enough to compute

$$\mathcal{F} = \frac{2}{s_0^3}\left(\mu_0 + \sum_{i=1}^{4} 2\mu_i \frac{(z_i-1)^2}{\prod_{i\neq j}(z_i - z_j)}\frac{\partial}{\partial z_i}\right) R_{\mathcal{F}} \tag{3.37}$$

with

$$R_{\mathcal{F}} = \frac{1}{2}\int_0^\infty \frac{du}{\sqrt{(z_1+u)(z_2+u)(z_3+u)(z_4+u)}} \tag{3.38}$$

and

$$\mu_i = \begin{cases} A' & \text{for } i = 0 \\ A' z_i^2 - (2A' + B')z_i + (A' + B' + C') & \text{for } 1 \leq i \leq 4 \end{cases} \tag{3.39}$$

where the parameters z_i with $1 \leq i \leq 4$ are roots of the function

$$z^4 - s_1 z^3 + s_2 z^2 - s_3 z + s_4 = 0 \tag{3.40}$$

with coefficients

$$\begin{aligned} s_1 &= \frac{E + 2F + 3G + 4H}{D + E + F + G + H}\,,\; s_2 = \frac{F + 3G + 6H}{D + E + F + G + H}\,, \\ s_3 &= \frac{G + 4H}{D + E + F + G + H}\,,\; s_4 = \frac{H}{D + E + F + G + H} \end{aligned} \tag{3.41}$$

that correspond to the elementary symmetric polynomials of the roots

$$s_1 = \sum_{1\leq i\leq 4} z_i\,, \quad s_2 = \sum_{1\leq i<j\leq 4} z_i z_j\,, \; s_3 = \sum_{1\leq i<j<k\leq 4} z_i z_j z_k\,, \; s_4 = z_1 z_2 z_3 z_4 \tag{3.42}$$

The complexity of the original elliptic integral, $\mathcal{F}$, has been reduced by describing it in terms of the symmetric integral, $R_{\mathcal{F}}$, and its derivatives. However, as it stands, the evaluation of the symmetric integral is still a difficult process. The next step is to further decompose the problem by describing the symmetric integral in a more-applicable manner.

3.2.2 Genus-One Curves and Their Jacobians

Consider the Jacobian variety of the smooth genus-one curve $\mathcal{C}$ in Eq. (3.26), $\mathrm{Jac}(\mathcal{C})$. It is an elliptic curve, $\mathcal{E}$, that can be represented, over $\mathbb{C}$, as a fully-factorized, plane cubic curve with affine coordinates $(\xi, \eta) \in \mathbb{C}^2$ of the form

$$\mathcal{E} \cong \mathrm{Jac}(\mathcal{C}): \quad \eta^2 = (K_2^2 + \xi)(K_3^2 + \xi)(K_4^2 + \xi) \tag{3.43}$$

Since the period described by the symmetric integral in Eq. (3.38) is a characteristic quantity of the Jacobian variety, $\mathrm{Jac}(\mathcal{C})$, rather than the genus-one curve, $\mathcal{C}$, it is possible to reduce the symmetric integral, $R_{\mathcal{F}}$, to a period integral for the elliptic curve $\mathcal{E}$ [41].

Assume that the discriminant of $\mathcal{C}$ never vanishes (i.e. $\Delta_{\mathcal{C}} \neq 0$). Then, setting $z_i = Z_i^2$ for $1 \leq i \leq 4$, where the parameters Z_i are located in the extended, open, right-half complex plane (i.e. $Z_1, ..., Z_4 \in \mathbb{C}_{\mathrm{Re}>0} \cup \{0\}$), and defining the new parameters

$$K_j = Z_1 Z_j + Z_k Z_l \quad \text{for}\, \{j,k,l\} = \{2,3,4\}, \{3,2,4\}, \{4,2,3\} \tag{3.44}$$

new rational functions $\xi, \eta \in \mathbb{C}(u, v)$ are defined on $\mathcal{C}$ by the expressions

$$\begin{aligned}\xi &= 2v + 2u^2 + (Z_1^2 + Z_2^2 + Z_3^2 + Z_4^2)u - 2Z_1Z_2Z_3Z_4 \\ \eta &= 4u^3 + 4uv + (Z_1^2 + Z_2^2 + Z_3^2 + Z_4^2)v + (Z_1^2Z_2^2Z_3^2 + Z_1^2Z_2^2Z_4^2 + Z_1^2Z_3^2Z_4^2 + Z_2^2Z_3^2Z_4^2) \\ &\quad + 2(Z_1^2Z_2^2 + Z_1^2Z_3^2 + Z_1^2Z_4^2 + Z_2^2Z_3^2 + Z_2^2Z_4^2 + Z_3^2Z_4^2)u + 3(Z_1^2 + Z_2^2 + Z_3^2 + Z_4^2)u^2\end{aligned} \tag{3.45}$$

It is easily verified that for $(u, v) = (0, 0)$ and $(u, v) = (\infty, \infty)$, these rational functions return $(\xi, \eta) = (0, 0)$ and $(\xi, \eta) = (\infty, \infty)$, respectively. In fact, the map $(u, v) \mapsto (\xi, \eta)$ defines a rational double cover, $\Phi : \mathcal{C} \dashrightarrow \mathcal{E}$, between the genus-one curve $\mathcal{C}$ and the elliptic curve $\mathcal{E}$. Furthermore, Eq. (3.45) induces an isomorphism $\mathrm{Jac}(\mathcal{C}) \cong \mathcal{E}$, because $\mathcal{C}_{\bar{k}} \cong \mathcal{E}_{\bar{k}}$ over the algebraic closure $\bar{k}$, where k is the function field of the moduli space. It readily follows that

$$\Phi^* \left(\frac{d\xi}{\eta} \right) = \frac{du}{v} \tag{3.46}$$

Based on the relation between the genus-one curve, $\mathcal{C}$, and its Jacobian variety, $\mathrm{Jac}(\mathcal{C})$, the symmetric integral in Eq. (3.38) can be rewritten

$$\begin{aligned} R_{\mathcal{F}} &= \frac{1}{2} \int_0^\infty \frac{du}{\sqrt{(Z_1^2 + u)(Z_2^2 + u)(Z_3^2 + u)(Z_4^2 + u)}} \\ &= \frac{1}{2} \int_0^\infty \frac{d\xi}{\sqrt{(K_2^2 + \xi)(K_3^2 + \xi)(K_4^2 + \xi)}} \end{aligned} \tag{3.47}$$

where the relations between Z_i and K_i are defined by Eq. (3.44) [41]. In this form, the symmetric integral can be evaluated using hypergeometric functions, leading to a closed-form solution to Eq. (3.20).

3.2.3 Relation to Hypergeometric Functions

The generalized hypergeometric function F_1, or *(first) Appell function*, is a formal extension of the Gauss hypergeometric function to two variables. For complex variables x_1, x_2 with $\max(|x_1|, |x_2|) < 1$, and rational parameters $\alpha, \beta_1, \beta_2 \in (0, 1) \cap \mathbb{Q}$ and $\gamma \in$

$(0,1]\cap\mathbb{Q}$, it has the absolutely convergent power series expansion given by

$$F_1\left(\begin{matrix}\alpha;\beta_2,\beta_2\\ \gamma\end{matrix}\middle|\, x_1,x_2\right)=\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\frac{(\alpha)_{m+n}(\beta_1)_m(\beta_2)_n}{(\gamma)_{m+n}m!n!}\,x_1^m x_2^n \tag{3.48}$$

where $(q)_k=\Gamma(q+k)/\Gamma(q)$ is the Pochhammer symbol for the rising factorial. Moreover, the function F_1 has an integral representation, for $\mathrm{Re}(\alpha)>0$ and $\mathrm{Re}(\gamma-\alpha)>0$, given by

$$F_1\left(\begin{matrix}\alpha;\beta_1,\beta_2\\ \gamma\end{matrix}\middle|\, x_1,x_2\right)=\frac{\Gamma(\gamma)}{\Gamma(\alpha)\Gamma(\gamma-\alpha)}\int_0^1\frac{w^{\alpha-1}(1-w)^{\gamma-\alpha-1}}{(1-x_1w)^{\beta_1}(1-x_2w)^{\beta_2}}\,dw \tag{3.49}$$

Restricting $x_2=0$, the classical Gauss hypergeometric function is regained,

$$F_1\left(\begin{matrix}\alpha;\beta_1,\beta_2\\ \gamma\end{matrix}\middle|\, x_1,0\right)={}_2F_1\left(\begin{matrix}\alpha,\beta_1\\ \gamma\end{matrix}\middle|\, x_1\right)=\sum_{m=0}^{\infty}\frac{(\alpha)_m(\beta_1)_m}{(\gamma)_m m!}\,x_1^m \tag{3.50}$$

If $\gamma-\alpha=1$, the Gauss hypergeometric function satisfies an important reduction identity—useful to reduce its defining parameters—given by

$${}_2F_1\left(\begin{matrix}\alpha+1,\beta+1\\ \alpha+2\end{matrix}\middle|\, x\right)=\frac{\alpha+1}{\beta x}\left(\frac{1}{(1-x)^{\beta}}-{}_2F_1\left(\begin{matrix}\alpha,\beta\\ \alpha+1\end{matrix}\middle|\, x\right)\right) \tag{3.51}$$

Using the integral representation (3.49), the derivatives of F_1 are immediately found to be

$$\begin{aligned}\frac{\partial^{m+n}}{\partial x_1^m\partial x_2^n}F_1\left(\begin{matrix}\alpha;\beta_1,\beta_2\\ \gamma\end{matrix}\middle|\, x_1,x_2\right)&\\ =\frac{(\alpha)_{m+n}(\beta_1)_m(\beta_2)_n}{(\gamma)_{m+n}}\,F_1\left(\begin{matrix}\alpha+m+n;\beta_1+m,\beta_2+n\\ \gamma+m+n\end{matrix}\middle|\, x_1,x_2\right)&\end{aligned} \tag{3.52}$$

A relation is then found between the symmetric integral from Eq. (3.47) and these hypergeometric functions. Using the transformation

$$x_1=1-\frac{K_2^2}{K_4^2}\,,\ x_2=1-\frac{K_3^2}{K_4^2}\,,\ w=\frac{K_4^2}{K_4^2+\xi}\,,\ y=-\frac{K_4\eta}{(K_4^2+\xi)^2} \tag{3.53}$$

and the parameters $\alpha = \beta_1 = \beta_2 = \frac{1}{2}$ and $\gamma = \alpha + 1$, the symmetric integral, $R_{\mathcal{F}}$ in Eq. (3.47), is rewritten in terms of F_1, creating the identity

$$R_{\mathcal{F}} = \frac{1}{K_4} F_1\left(\begin{matrix} \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \\ \frac{3}{2} \end{matrix} \middle| 1 - \frac{K_2^2}{K_4^2}, 1 - \frac{K_3^2}{K_4^2} \right) \tag{3.54}$$

where $\Gamma(\frac{3}{2})/\Gamma(\frac{1}{2}) = \frac{1}{2}$, or, equivalently,

$$R_{\mathcal{F}} = \frac{1}{Z_1 Z_4 + Z_2 Z_3} F_1\left(\begin{matrix} \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \\ \frac{3}{2} \end{matrix} \middle| 1 - \frac{(Z_1 Z_2 + Z_3 Z_4)^2}{(Z_1 Z_4 + Z_2 Z_3)^2}, 1 - \frac{(Z_1 Z_3 + Z_2 Z_4)^2}{(Z_1 Z_4 + Z_2 Z_3)^2} \right) \tag{3.55}$$

with derivatives found using Eq. (3.52)

$$\begin{aligned} \frac{\partial}{\partial Z_i} R_{\mathcal{F}} = {} & \frac{\partial}{\partial Z_i} \left(\frac{1}{Z_1 Z_4 + Z_2 Z_3} \right) F_1\left(\begin{matrix} \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \\ \frac{3}{2} \end{matrix} \middle| x_1, x_2 \right) \\ & + \frac{1}{Z_1 Z_4 + Z_2 Z_3} \left(\frac{1}{6} F_1\left(\begin{matrix} \frac{3}{2}; \frac{3}{2}, \frac{1}{2} \\ \frac{5}{2} \end{matrix} \middle| x_1, x_2 \right) \frac{\partial x_1}{\partial Z_i} + \frac{1}{6} F_1\left(\begin{matrix} \frac{3}{2}; \frac{1}{2}, \frac{3}{2} \\ \frac{5}{2} \end{matrix} \middle| x_1, x_2 \right) \frac{\partial x_2}{\partial Z_i} \right) \end{aligned} \tag{3.56}$$

These relations finally bring the original elliptic integral, from Eq. (3.20), into a form that can be evaluated for the general case.

Alternatively, any elliptic integral can be brought into one of three Legendre's canonical forms, usually denoted by

$$\mathrm{F}[\phi, k] = \int_0^\phi \frac{d\theta}{\Delta}, \quad \mathrm{E}[\phi, k] = \int_0^\phi \Delta \, d\theta, \quad \Pi[\phi, k, n] = \int_0^\phi \frac{d\theta}{\Delta(1 + n \sin^2 \theta)} \tag{3.57}$$

with $\Delta = \sqrt{1 - k^2 \sin^2 \theta}$. They are also called the incomplete elliptic integrals of the first, second, and third kind, respectively. The complete elliptic integrals are then recovered by

$$\mathrm{K}(k) = \mathrm{F}\left[\frac{\pi}{2}, k\right], \quad \mathrm{E}(k) = \mathrm{E}\left[\frac{\pi}{2}, k\right], \quad \Pi(k, n) = \Pi\left[\frac{\pi}{2}, k, n\right] \tag{3.58}$$

Substituting $\sin^2 \theta = \frac{K_4^2 - K_2^2}{K_4^2 + \xi}$ into Eq. (3.54) yields

$$R_{\mathcal{F}} = \frac{1}{\sqrt{K_4^2 - K_2^2}} \mathrm{F}\left[\sqrt{\frac{K_4^2 - K_2^2}{K_4^2}}, \sqrt{\frac{K_4^2 - K_2^2}{K_4^2 - K_3^2}} \right] \tag{3.59}$$

Expressions of this kind have been known in the literature [41]. However, the use of the multivariate hypergeometric function F_1 is far superior as it pertains to its implementation, and the analysis hereafter is based on Eq. (3.55). For example, the use of Eq. (3.48) naturally allows a perturbation expansion of the solution to the general, parabolic vortex segment.

3.2.4 Perturbation Expansion and Pencil of Elliptic Curves

It is heretofore shown that the general elliptic integral

$$\mathcal{F} = \int_0^1 \frac{At^2 + Bt + C}{\Big(Dt^4 + Et^3 + Ft^2 + Gt + H\Big)^{3/2}} dt \tag{3.60}$$

is computed by the expression of the form

$$\mathcal{F} = \frac{2(Z_1 Z_2 Z_3 Z_4)^3}{H^{3/2}} \left(\mu_0 + \sum_{i=1}^{4} \mu_i \frac{(Z_i^2 - 1)^2}{Z_i \prod_{j \neq i} (Z_i^2 - Z_j^2)} \frac{\partial}{\partial Z_i} \right) R_{\mathcal{F}} \tag{3.61}$$

where

$$\mu_i = \begin{cases} A + B + C & \text{for } i = 0 \\ (A + B + C) Z_i^4 - (B + 2C) Z_i^2 + C & \text{for } 1 \leq i \leq 4 \end{cases} \tag{3.62}$$

and $R_{\mathcal{F}}$ is the symmetric integral, that can be expressed in terms of the multivariate Appell hypergeometric function

$$R_{\mathcal{F}} = \frac{1}{Z_1 Z_4 + Z_2 Z_3} F_1 \left(\begin{matrix} \frac{1}{2}; \frac{1}{2}, \frac{1}{2} \\ \frac{3}{2} \end{matrix} \middle| 1 - \frac{(Z_1 Z_2 + Z_3 Z_4)^2}{(Z_1 Z_4 + Z_2 Z_3)^2}, 1 - \frac{(Z_1 Z_3 + Z_2 Z_4)^2}{(Z_1 Z_4 + Z_2 Z_3)^2} \right) \tag{3.63}$$

whose derivatives, $\frac{\partial}{\partial Z_i} R_{\mathcal{F}}$, are described by Eq. (3.56). The parameters Z_i^2 for $1 \leq i \leq 4$ are the roots of the function

$$\big(Z^2\big)^4 - s_1 \big(Z^2\big)^3 + s_2 \big(Z^2\big)^2 - s_3 \big(Z^2\big) + s_4 = 0 \tag{3.64}$$

with coefficients

$$\begin{aligned} s_1 &= \frac{E+2F+3G+4H}{D+E+F+G+H}\,,\ s_2 = \frac{F+3G+6H}{D+E+F+G+H}\,,\\ s_3 &= \frac{G+4H}{D+E+F+G+H}\,,\ s_4 = \frac{H}{D+E+F+G+H} \end{aligned} \tag{3.65}$$

that correspond to the elementary symmetric polynomials

$$s_1 = \sum_{1\le i\le 4} Z_i^2\,,\quad s_2 = \sum_{1\le i<j\le 4} Z_i^2 Z_j^2\,,\ s_3 = \sum_{1\le i<j<k\le 4} Z_i^2 Z_j^2 Z_k^2\,,\ s_4 = Z_1^2 Z_2^2 Z_3^2 Z_4^2 \tag{3.66}$$

Equations (3.61) through (3.66), therefore, result in a closed-form solution to Eq. (3.60) if the roots Z_i^2, with $1 \le i \le 4$, are known. These roots can be difficult to find analytically, in the general case, and thus must be found numerically or, as is hereafter explained, approximated by a perturbation expansion.

Consider the value of the elliptic integral $\mathcal{F}$ for a pencil, $\mathcal{C}_\epsilon$, of genus-one curves varying over a complex disc of radius one (i.e. $\epsilon \in \mathbb{C}$ with $|\epsilon| < 1$) having a singularity at $\epsilon = 0$. Roughly speaking, the desired pencil, $\mathcal{C}_\epsilon$, whose period integral interpolates between the integral for a linear vortex element and a parabolic one, is obtained by setting

$$\begin{aligned} Z_3 &= 1+\epsilon\zeta_3\,,\\ Z_4 &= 1+\epsilon\zeta_4\,, \end{aligned} \tag{3.67}$$

such that these roots Z_3^2 and Z_4^2 coincide, and Eq. (3.26) becomes singular, for $\epsilon = 0$ where ζ_3 and ζ_4 are complex numbers that will be determined presently. Thus, we have

$$\mathcal{C}_\epsilon:\quad v^2 = \big(Z_1^2+u\big)\big(Z_2^2+u\big)\big((1+\epsilon\zeta_3)^2+u\big)\big((1+\epsilon\zeta_4)^2+u\big) \tag{3.68}$$

where it is also assumed that the roots for $\epsilon = 1$ are pairwise different, and satisfy $Z_1, ..., Z_4 \in \mathbb{C}_{\mathrm{Re}>0}\cup\{0\}$. It is easily verified that the corresponding pencil of elliptic curves, $\mathcal{E}_\epsilon \cong \mathrm{Jac}(\mathcal{C}_\epsilon)$, is a smooth family for $0 < |\epsilon| < 1$, and has an isolated singular fiber of Kodaira type I_2 at $\epsilon = 0$ [42–44]. Using this pencil, a perturbation expansion for $\mathcal{F}$ is computed, with the

form

$$\mathcal{F} = \mathcal{F}^{(0)} + \epsilon\, \mathcal{F}^{(1)} + O(\epsilon^2) \tag{3.69}$$

Recall the parameterization of the parabolic vortex filament defined in Eq. (3.3). Defining,

$$\epsilon = |\vec{r}_0 - \vec{r}_1 - \vec{f}_0'| \tag{3.70}$$

the parabolic filament can be treated as a first-order perturbation of the linear vortex filament described in Eq. (3.7)

$$\vec{f}(t) = \epsilon \hat{r} t^2 + \vec{f}_0' t + \vec{f}_0 \tag{3.71}$$

where $\hat{r}$ is the unit vector

$$\hat{r} = \frac{\vec{r}_0 - \vec{r}_1 - \vec{f}_0'}{|\vec{r}_0 - \vec{r}_1 - \vec{f}_0'|} \tag{3.72}$$

such that $\vec{f}_0' = (\vec{r}_0 - \vec{r}_1) - \epsilon\hat{r}$. With such an expansion in ϵ, the asymptotic behavior of the polynomial coefficients in Eq. (3.60) turns out to be

$$A(\epsilon) = \epsilon(\vec{f}_0' \times \hat{r}) + O(\epsilon^2), \quad B(\epsilon) = 2\epsilon(\hat{r} \times \vec{r}_0) + O(\epsilon^2), \quad C(\epsilon) = \vec{f}_0' \times \vec{r}_0 + O(\epsilon^2),$$

$$D(\epsilon) = \epsilon^2 + O(\epsilon^3), \quad E(\epsilon) = 2\epsilon(\vec{f}_0' \cdot \hat{r}) + O(\epsilon^2), \quad F(\epsilon) = f_0'^2 - 2\epsilon(\vec{r}_0 \cdot \hat{r}) + O(\epsilon^2), \tag{3.73}$$

$$G(\epsilon) = -2\vec{f}_0' \cdot \vec{r}_0 + O(\epsilon^2), \quad H(\epsilon) = r_0^2 + O(\epsilon^2)$$

The integral in Eq. (3.60) can then be organized in terms of the integrals

$$\mathcal{F}_i = \int_0^1 \frac{t^i}{\left(Dt^4 + Et^3 + Ft^2 + Gt + H\right)^{3/2}} dt \quad \text{for } 0 \le i \le 2 \tag{3.74}$$

each with a perturbation expansion of the form

$$\mathcal{F}_i = \mathcal{F}_i^{(0)} + \epsilon\, \mathcal{F}_i^{(1)} + O(\epsilon^2) \tag{3.75}$$

Therefore, the perturbation expansion in Eq. (3.69) can be further decomposed as

$$\mathcal{F} = \underbrace{C(0)\,\mathcal{F}_0^{(0)}}_{\mathcal{F}^{(0)}} + \epsilon \underbrace{\left(C(0)\,\mathcal{F}_0^{(1)} + B'(0)\,\mathcal{F}_1^{(0)} + A'(0)\,\mathcal{F}_2^{(0)} \right)}_{\mathcal{F}^{(1)}} + O(\epsilon^2) \tag{3.76}$$

with $A'(0) = \frac{dA}{d\epsilon}|_{\epsilon=0}$ and $B'(0) = \frac{dB}{d\epsilon}|_{\epsilon=0}$.

In the case that $\epsilon = 0$, the coefficients in Eq. (3.73) can be seen to match those of the linear vortex case described by Eq. (3.8). Thus, when determining $\mathcal{F}^{(0)}$ and $\mathcal{F}^{(1)}$ from Eq. (3.69), $\mathcal{F}^{(0)}$ is by construction the solution to the influence of a linear vortex filament, Eq. (3.9), and $\mathcal{F}^{(1)}$ is a first-order approximation of the parabolic effects. To start, consider a pencil

$$\mathcal{C}_\epsilon : \quad v^2 = \big(Z_1(\epsilon)^2 + u\big)\big(Z_2(\epsilon)^2 + u\big)\big(Z_3(\epsilon)^2 + u\big)\big(Z_4(\epsilon)^2 + u\big) \tag{3.77}$$

with the roots, $Z_i = Z_i(\epsilon)$ for $1 \le i \le 4$, given by

$$\begin{aligned}
Z_1 &= \frac{\sqrt{\vec{r}_0 \cdot \vec{r}_1 + r_0 r_1} + \sqrt{\vec{r}_0 \cdot \vec{r}_1 - r_0 r_1}}{\sqrt{2} r_1} - \epsilon \frac{\sqrt{2}}{2} \frac{Z_N^{(1)}}{Z_D} \\
Z_2 &= \frac{\sqrt{\vec{r}_0 \cdot \vec{r}_1 + r_0 r_1} - \sqrt{\vec{r}_0 \cdot \vec{r}_1 - r_0 r_1}}{\sqrt{2} r_1} + \epsilon \frac{\sqrt{2}}{2} \frac{Z_N^{(2)}}{r_1^3 Z_D} \\
Z_3 &= 1 + \frac{\epsilon}{2} \frac{\vec{f}_0' \cdot \hat{r} + \sqrt{2(\vec{r}_0 \cdot \vec{r}_1) + (\vec{f}_0' \cdot \hat{r})^2 - r_0^2 - r_1^2}}{2(\vec{r}_0 \cdot \vec{r}_1) - r_0^2 - r_1^2} \\
Z_4 &= 1 - \frac{\epsilon}{2} \frac{1}{\vec{f}_0' \cdot \hat{r} + \sqrt{2(\vec{r}_0 \cdot \vec{r}_1) + (\vec{f}_0' \cdot \hat{r})^2 - r_0^2 - r_1^2}}
\end{aligned} \tag{3.78}$$

The quantities $Z_N^{(i)}$ and Z_D with $i = 1, 2$ in Eq. (3.78) are then obtained by fine-tuning the elliptic pencil so the resulting elliptic integral matches the asymptotic behavior given by

Eq. (3.73), resulting in the expressions

$$\begin{aligned}
Z_N^{(1)} =& r_0^3\left((\vec{f_0}\cdot\hat{r})\sqrt{(\vec{r}_0\cdot\vec{r}_1)^2-r_0^2r_1^2}+(\vec{r}_1\cdot\hat{r})r_0^2+(\vec{r}_0\cdot\hat{r})r_1^2-(\vec{r}_0\cdot\vec{r}_1)\Big((\vec{r}_0+\vec{r}_1)\cdot\hat{r}\Big)\right)\\
Z_N^{(2)} =& \Bigg(\Big(-r_0^4r_1^2+4(\vec{r}_0\cdot\vec{r}_1)r_0^2r_1^2+4(\vec{r}_0\cdot\vec{r}_1)^2r_0^2-8(\vec{r}_0\cdot\vec{r}_1)^3\Big)(\vec{f_0}\cdot\hat{r})\\
&-(4\vec{r}_0\cdot\vec{r}_1-r_0^2r_1^2)(\vec{r}_0-\vec{r}_1)^2(\vec{r}_0\cdot\hat{r})\Bigg)\sqrt{\vec{r}_0\cdot\vec{r}_1+r_0r_1}\sqrt{\vec{r}_0\cdot\vec{r}_1-r_0r_1}\\
&+\left(r_0^4r_1^4+3(\vec{r}_0\cdot\vec{r}_1)r_0^4r_1^2-8(\vec{r}_0\cdot\vec{r}_1)^2r_0^2r_1^2-4(\vec{r}_0\cdot\vec{r}_1)^3r_0^2+8(\vec{r}_0\cdot\vec{r}_1)^4\right)(\vec{f_0}\cdot\hat{r})\\
&+\Big(4(\vec{r}_0\cdot\vec{r}_1)^2-3r_0^2r_1^2\Big)(\vec{r}_0-\vec{r}_1)^2(\vec{r}_0\cdot\vec{r}_1)(\vec{r}_0\cdot\hat{r})\\
Z_D =& \big(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2\big)\Big((\vec{r}_0\cdot\vec{r}_1-r_0r_1)(2\vec{r}_0\cdot\vec{r}_1+r_0r_1)\sqrt{\vec{r}_0\cdot\vec{r}_1+r_0r_1}\\
&-(\vec{r}_0\cdot\vec{r}_1+r_0r_1)(2\vec{r}_0\cdot\vec{r}_1-r_0r_1)\sqrt{\vec{r}_0\cdot\vec{r}_1-r_0r_1}\Big)
\end{aligned}$$

With these roots and using equations (3.44), an expansion of Eq. (3.53) of the form $x_i = x_i^{(0)} + \epsilon x_i^{(1)} + O(\epsilon^2)$ for $1 \leq i \leq 2$, yields

$$x_1^{(0)} = \frac{2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}{2(\vec{r}_0\cdot\vec{r}_1+r_0r_1)}\,, \qquad x_2^{(0)} = 0 \tag{3.79}$$

This shows why expressing the elliptic integral in terms of the multivariate hypergeometric function F_1 is a powerful analytical tool: in the limit $\epsilon \to 0$ one finds $x_2 = 0$ and the multivariate hypergeometric function F_1 restricts to the Gauss hypergeometric function in Eq. (3.50).

Carrying out the series expansion of Eq. (3.61), the leading order term, $\mathcal{F}^{(0)}$, is found to be

$$\begin{aligned}
\mathcal{F}^{(0)} =& \frac{2C(0)\,Z_1^2Z_2^2(1+Z_1Z_2)^2}{H^{3/2}(Z_1+Z_2)^3}F_1\left(\begin{matrix}\frac{1}{2};\frac{1}{2},\frac{1}{2}\\ \frac{3}{2}\end{matrix}\,\middle|\, x_1^{(0)},0\right)\\
&-\frac{4C(0)\,Z_1^2Z_2^2(1+Z_1Z_2)^2(1-Z_1^2)(1-Z_2^2)}{H^{3/2}(Z_1+Z_2)^5}\frac{\partial}{\partial x_1}F_1\left(\begin{matrix}\frac{1}{2};\frac{1}{2},\frac{1}{2}\\ \frac{3}{2}\end{matrix}\,\middle|\, x_1^{(0)},0\right)+O(\epsilon)
\end{aligned} \tag{3.80}$$

Using equations (3.50) – (3.52), and equations (3.78) – (3.79), this term can be rewritten to yield

$$\mathcal{F}^{(0)} = \frac{2C(0)\,Z_1^2 Z_2^2(1+Z_1Z_2)}{H^{3/2}(Z_1+Z_2)^2} + O(\epsilon) = \frac{C(0)\,(r_0+r_1)}{(\vec{r}_0\cdot\vec{r}_1+r_0r_1)r_0r_1} \tag{3.81}$$

matching precisely Eq. (3.9), as designed. A similar computation is repeated for each of the remaining terms in Eq. (3.76), resulting in the set of expressions

$$\begin{aligned}
\mathcal{F}_0^{(0)} &= \frac{r_0+r_1}{r_0r_1(\vec{r}_0\cdot\vec{r}_1+r_0r_1)} \\
\mathcal{F}_0^{(1)} &= \frac{(\vec{r}_0\cdot\hat{r})(\vec{r}_0\cdot\vec{r}_1+2r_0r_1+r_1^2)}{r_1^3(\vec{r}_0\cdot\vec{r}_1+r_0r_1)^2} - \frac{(\vec{f}_0'\cdot\hat{r})(\vec{r}_0\cdot\vec{r}_1+2r_0r_1)}{r_1^3(\vec{r}_0\cdot\vec{r}_1+r_0r_1)^2} \\
&\quad - \frac{(r_0+r_1)\sqrt{(\vec{f}_0'\cdot\hat{r})^2+2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}}{(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2)\sqrt{\vec{r}_0\cdot\vec{r}_1-r_0r_1}(\vec{r}_0\cdot\vec{r}_1+r_0r_1)^{3/2}} \\
&\quad + \frac{2\sqrt{2}\sqrt{(\vec{f}_0'\cdot\hat{r})^2+2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}}{(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2)(\vec{r}_0\cdot\vec{r}_1+r_0r_1)\sqrt{\vec{r}_0\cdot\vec{r}_1-r_0r_1}}\,{}_2F_1\left(\begin{matrix}\frac{1}{2},-\frac{1}{2}\\ \frac{3}{2}\end{matrix}\middle|\,\frac{2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}{2(\vec{r}_0\cdot\vec{r}_1+r_0r_1)}\right) \\
&\quad - \frac{\sqrt{2}\,\sqrt{(\vec{f}_0'\cdot\hat{r})^2+2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}}{(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2)(\vec{r}_0\cdot\vec{r}_1+r_0r_1)\sqrt{\vec{r}_0\cdot\vec{r}_1-r_0r_1}}\,{}_2F_1\left(\begin{matrix}\frac{1}{2},\frac{1}{2}\\ \frac{3}{2}\end{matrix}\middle|\,\frac{2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}{2(\vec{r}_0\cdot\vec{r}_1+r_0r_1)}\right) \\
\mathcal{F}_1^{(0)} &= \frac{1}{r_1(\vec{r}_0\cdot\vec{r}_1+r_0r_1)} \\
\mathcal{F}_2^{(0)} &= \frac{(2\vec{r}_0\cdot\vec{r}_1+r_0(r_1-r_0))}{r_1(\vec{r}_0\cdot\vec{r}_1+r_0r_1)(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2)} \\
&\quad - \frac{\sqrt{2}}{(2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2)\sqrt{\vec{r}_0\cdot\vec{r}_1+r_0r_1}}\,{}_2F_1\left(\begin{matrix}\frac{1}{2},\frac{1}{2}\\ \frac{3}{2}\end{matrix}\middle|\,\frac{2\vec{r}_0\cdot\vec{r}_1-r_0^2-r_1^2}{2(\vec{r}_0\cdot\vec{r}_1+r_0r_1)}\right)
\end{aligned} \tag{3.82}$$

that, when used with Eq. (3.82) in Eq. (3.76), provides an approximation of Eq. (3.60) up to $O(\epsilon^2)$.

3.3 Validation and Comparison of Predictive Methods

Consider five methods of evaluating the integral

$$\vec{V} = \frac{\Gamma}{4\pi}\int_0^1 \frac{\Big(At^2+Bt+C\Big)dt}{\Big(Dt^4+Et^3+Ft^2+Gt+H\Big)^{3/2}} \tag{3.83}$$

where

$$A = \vec{f}_0' \times (\vec{r}_0 - \vec{r}_1), \quad B = -2(\vec{r}_1 + \vec{f}_0') \times \vec{r}_0, \quad C = \vec{f}_0' \times \vec{r}_0,$$

$$D = (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')^2, \quad E = 2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0') \cdot \vec{f}_0', \quad F = f_0'^2 - 2\vec{r}_0 \cdot (\vec{r}_0 - \vec{r}_1 - \vec{f}_0'),$$

$$G = -2\vec{r}_0 \cdot \vec{f}_0', \quad H = r_0^2$$

First, in Section 3.2, Eq. (3.83) is related to hypergeometric functions, resulting in the closed-form expression

$$\vec{V} = \frac{\Gamma}{4\pi} \frac{2(Z_1 Z_2 Z_3 Z_4)^3}{H^{3/2}} \left(\mu_0 + \sum_{i=1}^{4} \mu_i \frac{(Z_i^2 - 1)^2}{Z_i \prod_{j \neq i} (Z_i^2 - Z_j^2)} \frac{\partial}{\partial Z_i} \right) R_{\mathcal{F}} \tag{3.84}$$

where equations (3.62) – (3.66) describe μ_i, $R_{\mathcal{F}}$, and Z_i. Eq. (3.84) results in a closed-form solution to Eq. (3.83) if the roots Z_i^2, with $1 \leq i \leq 4$, can be readily found.

Second, in Section 3.2.4, Eq. (3.84) is expanded in ϵ, leading to the approximation

$$\vec{V} = \frac{\Gamma}{4\pi} \Big(C(0)\, \mathcal{F}_0^{(0)} + \epsilon \Big(C(0)\, \mathcal{F}_0^{(1)} + B'(0)\, \mathcal{F}_1^{(0)} + A'(0)\, \mathcal{F}_2^{(0)} \Big) + O(\epsilon^2) \Big) \tag{3.85}$$

where equations (3.70) and (3.73) define ϵ, A, B, and C, and Eq. (3.82) defines $\mathcal{F}_i^{(j)}$. This expansion is about the case of a vortex line segment (i.e. $\epsilon = 0$), and thus the best results can be expected when the parabolic vortex segment has little curvature.

Third, discussed in Section 3.1.1.2, is an approximation made in Bliss et al. for the case of the parabolic parameterization [11]

$$\begin{aligned} \vec{f}(t) &= (\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t^2 + \vec{f}_0' t + \vec{f}_0 \\ d\vec{f}(t) &= \big(2(\vec{r}_0 - \vec{r}_1 - \vec{f}_0')t + \vec{f}_0'\big) dt \end{aligned} \tag{3.86}$$

where

$$\vec{f}_0 = \big[-\ell, \varepsilon \ell^2, 0 \big]$$

$$\vec{r}_0 - \vec{r}_1 = \big[2\ell, 0, 0 \big] \tag{3.87}$$

$$\vec{f_0'} = \left[2\ell, -4\varepsilon\ell^2, 0\right]$$

in which Eq. (3.83) is approximated by the integral

$$\vec{V} = \frac{\Gamma}{4\pi}\int_{-\ell}^{\ell} \frac{\left[2\varepsilon z_0 x,\ -z_0,\ -2\varepsilon x x_0 + \varepsilon x^2 + y_0\right]dx}{\left((1-2\varepsilon y_0+\varepsilon^2 F_2)x^2 + (-2x_0+\varepsilon^2 F_1)x + (x_0^2+y_0^2+z_0^2+\varepsilon^2 F_0)\right)^{3/2}} \quad (3.88)$$

where F_2, F_1, and F_0 must be tuned to accurately replicate the original integral. Note the difference between ε, used by Bliss et al. and defined in Eq. (3.11), and ϵ, used in the perturbation expansion in Eq. (3.85) and defined in Eq. (3.70).

Fourth, Eq. (3.83) can be evaluated numerically. Numerical integration can be an effective means of evaluating an integral, so long as the integral is subdivided into a sufficient number of sections, and the integral itself does not exhibit highly oscillatory or asymptotic behavior. In the case of Eq. (3.83), the denominator tends to zero as the point at which the induced velocity is calculated moves close to the vortex filament, thus, accurate numerical results may be expected away from the vortex, but the results may lose fidelity as the vortex is approached.

Finally, the results of the first four methods can be compared to the approximation of the parabolic vortex segment by several of the straight vortex segments described in section 3.1.1.1. Such a comparison will provide insight into the number of straight segments needed to accurately reproduce the parabolic segment, and perhaps hint at the advantages provided by the use of parabolic vortices.

The comparisons made in this work will be restricted to a planar problem in which the point of interest resides in the same plane as a parabolic vortex segment of the form

$$\vec{f_0} = \left[-\ell, \varepsilon\ell^2, 0\right]$$

$$\vec{r_0} - \vec{r_1} = \left[2\ell, 0, 0\right] \quad (3.89)$$

$$\vec{f_0'} = \left[4\kappa + 2\ell, -4\varepsilon\ell^2, 0\right]$$

which is a simple extension of Eq. (3.87) that allows for the description of asymmetric, parabolic vortices when $\kappa \neq 0$. Sample points will be taken along the y-axis ($x = 0$). Additionally, all comparisons will assume $\Gamma = 4\pi$, to further reduce the degrees of freedom under consideration.

3.3.1 Analytic vs Numerical Integration

To validate the analytic solution derived herein, Eq. (3.83) is numerically integrated, using Simpson's rule, and compared to the results of Eq. (3.84). Numerical validation of this nature is manifest when the predictions obtained by numerical integration approach those obtained using the analytic formula as the number of intervals used in the integration increases towards infinity. The comparison between numerical integration and the analytic solution is made both for a symmetric case with little curvature, $\{\varepsilon, \kappa\} = \{-0.01, 0.0\}$, and an asymmetric case with larger curvature, $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$. Each of these cases is predicted using three different quantities of intervals, $n = \{10, 20, 40\}$, to determine numerical convergence and identify the relationship between the number of intervals used and the accuracy of the prediction. The results are shown in Fig. 3.2.

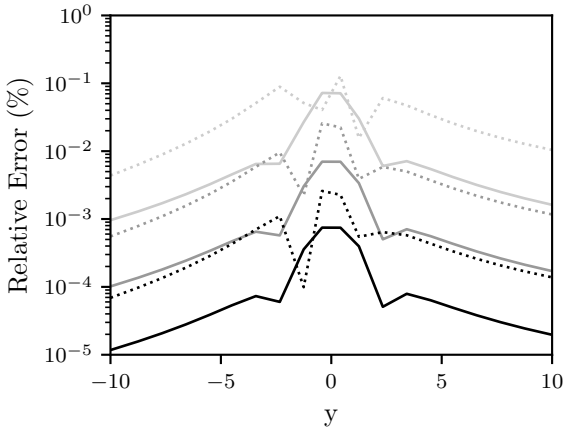

Fig. 3.2: The relative error of the numerical integration with respect to the analytic solution expressed in Eq. (3.84), for $\{\varepsilon, \kappa\} = \{-0.01, 0.0\}$ (solid) and $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$ (dashed). Each line represents a different number of intervals used in the numerical integration, $n = \{10, 20, 40\}$ (lightest to darkest).

It can seen in Fig. 3.2 that the numerical integration is convergent for the two cases displayed, observing that the error diminishes linearly on a logarithmic scale as the number of intervals increases. And, as predicted, the error is higher close to the vortex ($y \approx 0$), and decreases further away. It is also observed that the relative error of the numerical integration is larger for the asymmetric, high-curvature case than for the symmetric, low-curvature case. This suggests that a higher number of intervals is required as the vortex segment becomes increasingly dissimilar to a linear segment.

3.3.2 Analytic vs Approximation using Straight Segments

To further corroborate the results obtained from the explicit formula in Eq. (3.84) and numerical integration, predictions are made by approximating the parabolic vortex segment with several linear vortex segments. Again, the comparison is made for a symmetric case with little curvature, $\{\varepsilon, \kappa\} = \{-0.01, 0.0\}$, and an asymmetric case with larger curvature, $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$, and each case is predicted using three quantities of linear vortex segments, $n = \{10, 20, 40\}$. The results are shown in Fig. 3.3.

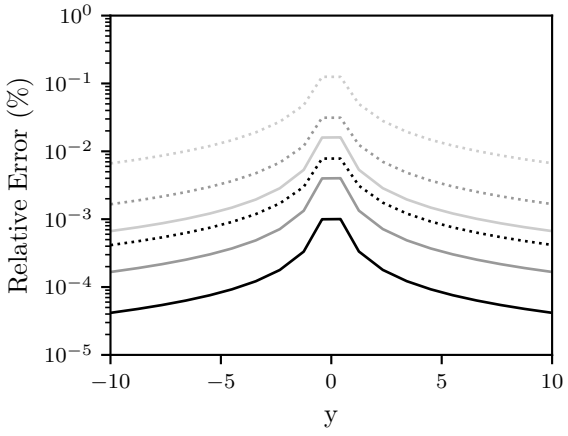

Fig. 3.3: The relative error of the approximation of the vortex with several straight vortex segments with respect to the analytic solution expressed in Eq. (3.84), for $\{\varepsilon, \kappa\} = \{-0.01, 0.0\}$ (solid) and $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$ (dashed). Each line represents a different number of linear segments used, $n = \{10, 20, 40\}$ (lightest to darkest).

As with the results from numerical integration, it can seen in Fig. 3.3 that the linear-segment approximation is convergent for the two cases displayed, and that the relative error is larger for the asymmetric, high-curvature case than for the symmetric, low-curvature case. These cases suggest that at least ten linear vortex segments are required to represent a single parabolic vortex segment within 0.1%. It is noted that the relative error of the linear-vortex approximation is, in general, higher than that of numerical integration, for an equal number of intervals and segments, though the relative error of both methods is well under 0.1% for the cases observed.

3.3.3 Analytic vs Approximation by Bliss et al.

Having validated the analytic solution using two computational methods, a comparison is performed between the approximation proposed by Bliss et al. in Eq. (3.88) and the explicit formula expressed in Eq. (3.84). The comparison is made with three values of ϵ (from -0.01 to -1.0) for both a symmetric ($\kappa = 0.0$) and an asymmetric ($\kappa = 0.5$) parabolic vortex. The results are shown in Fig. 3.4.

Again, the results of this comparison look as expected—the error of the approximation diminishes as ϵ tends to zero. The case of the symmetric vortex results in less error than

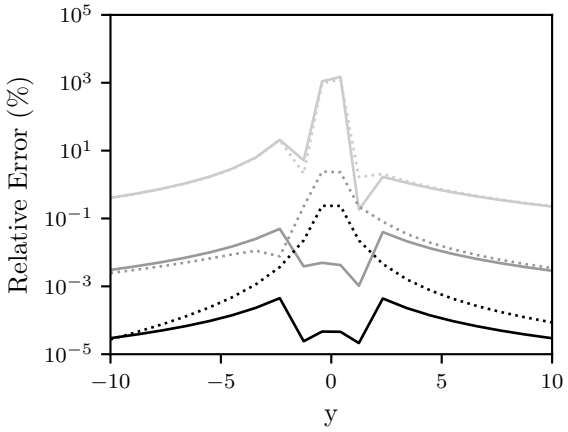

Fig. 3.4: The relative error of the approximation by Bliss et al. in Eq. (3.88) with respect to the analytic solution expressed in Eq. (3.84). Each line represents a different value of ε and κ, where $\varepsilon = \{-0.01, -0.1, -1.0\}$(darkest to lightest) and $\kappa = \{0.0, 0.5\}$ (solid, dashed).

the asymmetric vortex, close to the vortex itself ($y \approx 0$), through the relative error for both cases is similar far away from the vortex, suggesting that the effect of the asymmetry on the induced velocity far from the vortex is negligible. Similarly, as ϵ grows larger—such as the case that $\epsilon = -1.0$ in Fig. 3.4—the effect of asymmetry on the relative error of the approximation also appears to become insignificant. This is perhaps because the error due to the large curvature overshadows and contribution to the error from the asymmetry.

3.3.4 Analytic vs Perturbation Expansion

It is now of interest to compare the analytic solution expressed in Eq. (3.84) with its perturbation expansion, expressed in Eq. (3.85). The comparison is made by simultaneously varying ε from -0.001 to -0.09 and κ from 0.0 to 0.5 in order to vary ϵ from 0.004 to 2.0, as defined by Eq. (3.70). The results are shown in Fig. 3.5.

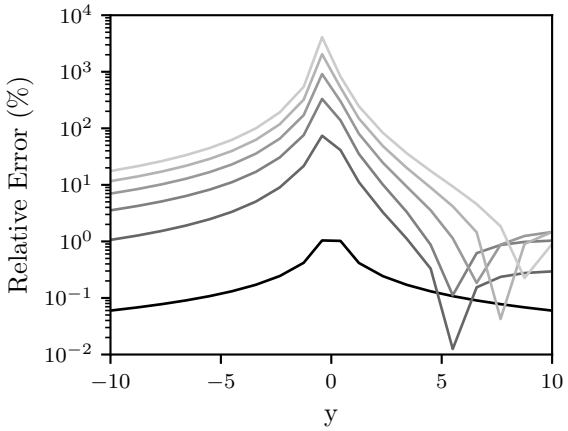

Fig. 3.5: The relative error of the perturbation expansion in Eq. (3.85) with respect to the analytic solution expressed in Eq. (3.84). Each line represents a different value of ϵ (created by varying ε and κ), such that $\epsilon = \{0.004, 0.4, 0.8, 1.2, 1.6, 2.0\}$ (darkest to lightest).

As expected, the error of the perturbation expansion, relative to the analytical solution, decreases as ϵ approaches zero. Unfortunately, in order for the $O(\epsilon^2)$ expansion described in Eq. (3.85) to approximate the analytical solution to within 1%, ϵ must remain prohibitively small for engineering applications. The inclusion of more terms in the expansion would

generalize the use of the approximation, but each term is increasingly more mathematically complex to derive.

3.3.5 Comparison of Computational Cost

In addition to comparisons of accuracy, it is of practical interest to compare the computational cost of each prediction method. Table 3.1 shows a comparison of the time required by the five methods discussed previously to predict the influence of the vortex at various distances along the y-axis. Forty divisions were used in both the numerical integration and linear-segments approximation. Each of the methods was naively implemented in Python, using intrinsic functions and libraries, and without substantial optimization for efficiency. The case wherein $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$ was selected to more clearly demonstrate the increase in computation cost required to compute the analytic prediction near the vortex itself.

Table 3.1: Time (in seconds) required to predict the influence of the parabolic vortex segment defined by $\{\varepsilon, \kappa\} = \{-0.1, 0.5\}$, by various methods at several locations along the y-axis.

y ($x = 0$)	Analytic (Eq. (3.84))	Perturbation (Eq. (3.85))	Bliss et al. (Eq. (3.88))	Numerical Integration	Linear Segments
-10.00	3.90E-2	8.53E-4	7.90E-5	2.21E-4	1.85E-4
-8.75	4.05E-2	5.36E-4	7.60E-5	1.90E-4	1.68E-4
-7.67	4.28E-2	5.50E-4	7.70E-5	2.23E-4	1.99E-4
-6.58	4.59E-2	5.58E-4	7.90E-5	2.39E-4	1.77E-4
-5.50	4.98E-2	5.41E-4	7.60E-5	1.89E-4	1.65E-4
-4.50	5.67E-2	6.30E-4	1.10E-4	2.21E-4	1.75E-4
-3.42	6.96E-2	5.48E-4	8.50E-5	1.86E-4	1.72E-4
-2.33	1.51E-1	5.78E-4	8.90E-5	2.14E-4	1.77E-4
-1.25	2.50E+0	8.03E-4	1.68E-4	2.26E-4	1.80E-4
-0.42	1.15E+0	5.00E-3	1.65E-4	2.11E-4	1.75E-4

From Table 3.1, it can be observed that the evaluation of the full analytic solution comes at a substantially higher computational cost than that of the other methods. This is due to the cost of computing the multivariate Appell hypergeometric function, and finding the roots of a quartic polynomial. The cost decreases drastically for the the perturbation expansion, where a hypergeometric function in only one variable is computed. The remaining three

methods require roughly the same time to compute, though the approximation by Bliss et al. is the fastest in all cases.

3.4 Conclusion

An explicit formula has been developed using multivariate hypergeometric functions to predict the velocity induced by a general parabolic vortex segment, summarized by Eq. (3.84). This formula is based on the integral that results from the Biot-Savart law, and is derived by constructing a genus-one curve whose period integrals provide the solution to the induced velocity. Moving from the genus-one curve to the corresponding Jacobian elliptic curve, the resulting elliptic integral can be evaluated explicitly using multivariate special functions. The evaluation of the formula in Eq. (3.84), through analytically explicit, requires finding the roots of a quartic polynomial and the implementation of the multivariate first Appell hypergeometric function, which complicates its practical implementation. Using the carefully crafted pencil of genus-one curves in Eq. (3.77), the series expansion of the first Appell hypergeometric function was used to derive a quadratic perturbation expansion to interpolate between the linear and the parabolic vortex segment. Equation (3.84) is validated through comparison to the predictions resulting from computational methods—numerical integration and the approximation of the parabolic vortex by several linear vortex segments. Figs. 3.2 and 3.3 show that those numerical methods converge to the analytic formula as the number of intervals or segments increase towards infinity, corroborating the validity of the explicit formula.

The parabolic vortex segments discussed in this chapter could be applied to the general implementation of lifting-line theory, approximating the bound vortex filament and replacing the finite segment of the jointed trailing vortices. However, the implementation of the explicit formula in Eq. (3.84) requires a higher computational cost than the cost required by the other methods discussed herein, especially when computing predictions close to the vortex, as is necessary in lifting-line theory. To this end, a perturbation expansion of the explicit formula, Eq. (3.85), was derived. However, despite the decrease in computational cost afforded by the expansion, the vortex must maintain a prohibitively small amount of

curvature for the expansion to be reasonably accurate. Therefore, the main advantage of the explicit formula derived in this chapter is that of analytic manipulation and academic insight into the relation between hypergeometric functions and the mechanics of a parabolic vortex, and it will not be applied to the general implementation of lifting-line theory in this body of work.

CHAPTER 4

MODELS OF THE LOCUS OF AERODYNAMIC CENTERS

The geometries of the bound vortex filament and trailing vortex sheet described in Chapter 2 are dependent on the shape of the locus of aerodynamic centers for a particular wing. The aerodynamic center of an airfoil is defined as the point about which the aerodynamic moment is invariant to small changes in angle of attack [3, 5]. For a finite wing, then, the locus of aerodynamic centers is the collection of points defining the location of the aerodynamic center of each spanwise section along the wing. For a planar wing without sweep, the locus of aerodynamic centers is well approximated by the *quarter-chord line* of the wing (i.e. the locus of points along a wing at 1/4 the distance from the leading edge to the trailing edge of the wing). When a wing is swept, the locus of aerodynamic centers becomes curved at the root and tip.

4.1 Küchemann's Approximation

In his 1956 paper, Küchemann modeled the shift in aerodynamic center from the quarter-chord along a wing of constant sweep, taking into account aspect ratio effects [28]. For a wing of large aspect ratio, the deviation of the aerodynamic center varies from

$$\frac{\Lambda}{2\pi}c \tag{4.1}$$

at the center and tip of the wing, where Λ is the sweep of the wing and c is the local chord length, to zero along the mid-span of the wing, as shown in Fig. 4.1. Küchemann interpolates between these values, using a hyperbola, to model the locus of aerodynamic centers for all points along the wing span. This interpolation results in the equation

$$f(z) = \frac{1}{4}c(0) + |z|\tan\Lambda + \lambda(\Lambda, z)\frac{\Lambda}{2\pi}c(z) \tag{4.2}$$

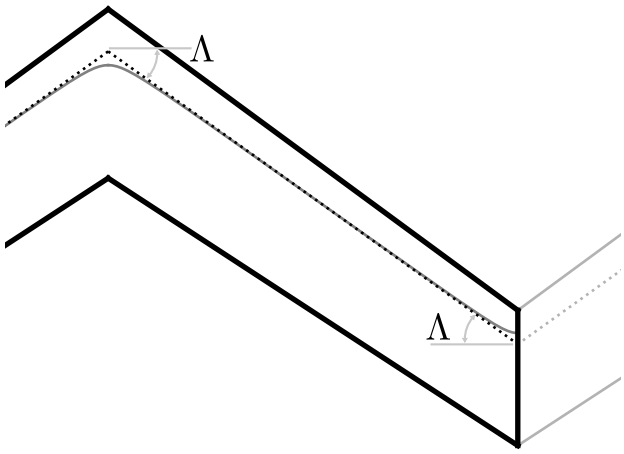

Fig. 4.1: The predicted locus of aerodynamic centers of a wing of large aspect ratio with constant sweep (not to scale).

where $\lambda(\Lambda, z)$ is the hyperbolic interpolation function

$$\begin{aligned}\lambda(\Lambda, z) =& \left(\sqrt{1 + \left(2\pi \frac{\tan\Lambda}{\Lambda} \frac{z}{c(z)} \right)^2} - 2\pi \frac{\tan\Lambda}{\Lambda} \frac{|z|}{c(z)} \right)_{\text{center}} \\ & - \left(\sqrt{1 + \left(2\pi \frac{\tan\Lambda}{\Lambda} \frac{\frac{b}{2} - |z|}{c(z)} \right)^2} - 2\pi \frac{\tan\Lambda}{\Lambda} \frac{\frac{b}{2} - |z|}{c(z)} \right)_{\text{tip}}\end{aligned} \tag{4.3}$$

The first term on the right hand side of Eq. (4.3) describes the effect of the wing center on the locus of aerodynamic centers, and the second term describes the effect of the wing tip. Note that, in Eq. (4.3), Küchemann treats the wing tip as the center of a wing with the opposite amount of sweep, see Fig. 4.1.

To account for the effects of aspect ratio, Küchemann defines an effective wing sweep angle

$$\Lambda_K = \frac{\Lambda}{\left(1 + \left(\frac{\tilde{C}_{L,\alpha} \cos\Lambda}{\pi R_A} \right)^2 \right)^{1/4}} \tag{4.4}$$

where $\tilde{C}_{L,\alpha}$ is the lift slope of the root airfoil, and R_A is the aspect ratio of the wing. Küchemann also adjusts Eq. (4.2), resulting in the function

$$f(z) = \frac{1}{4}c(0) + |z|\tan\Lambda_w - \frac{c(z)}{4}\left(1 - \frac{1}{K}\left(1 + 2\lambda(\Lambda_K, z)\frac{\Lambda_K}{\pi}\right)\right) \tag{4.5}$$

where

$$K = \left(1 + \left(\frac{\tilde{C}_{L,\alpha}\cos\Lambda_K}{\pi R_A}\right)^2\right)^{\frac{\pi}{4(\pi+2|\Lambda_K|)}} \tag{4.6}$$

For use in the general implementation of lifting-line theory, the first derivative of the locus of aerodynamic centers must be known to calculate the effective locus of aerodynamic centers as well as the jointed trailing vortices. The first derivative of Eq. (4.5) is

$$f'(z) = \frac{z}{|z|}\tan\Lambda_w + \lambda'(\Lambda_K, z)\frac{\Lambda_K}{2\pi}\frac{c(z)}{K} - \frac{c'(z)}{4}\left(1 - \frac{1}{K}\left(1 + 2\lambda(\Lambda_K, z)\frac{\Lambda_K}{\pi}\right)\right) \tag{4.7}$$

where

$$\begin{aligned}\lambda'(\Lambda_K, z) = &\left(\frac{4\pi^2\frac{\tan^2\Lambda_K}{\Lambda_K^2}\left(zc(z) - z^2c'(z)\right)}{c(z)^3\sqrt{1 + \left(2\pi\frac{\tan\Lambda_K}{\Lambda_K}\frac{z}{c(z)}\right)^2}} - 2\pi\frac{\tan\Lambda_K}{\Lambda_K}\frac{\frac{z}{|z|}c(z) - |z|c'(z)}{c(z)^2}\right)_{\text{center}} \\ &+ \left(\frac{4\pi^2\frac{\tan^2\Lambda_K}{\Lambda_K^2}\left(\frac{z}{|z|}(\frac{b}{2} - |z|)c(z) + (\frac{b}{2} - |z|)^2c'(z)\right)}{c(z)^3\sqrt{1 + \left(2\pi\frac{\tan\Lambda_K}{\Lambda_K}\frac{\frac{b}{2} - |z|}{c(z)}\right)^2}}\right. \\ &\left. - 2\pi\frac{\tan\Lambda_K}{\Lambda_K}\frac{\frac{z}{|z|}c(z) + (\frac{b}{2} - |z|)c'(z)}{c(z)^2}\right)_{\text{tip}}\end{aligned} \tag{4.8}$$

The results of lifting-line theory are likely dependent on the model used for the locus of aerodynamic centers. Fortunately, Eq. (4.5) has been proven sufficiently accurate by Moorthamers and Hunsaker [45], and such sensitivity is not to be explored in this work.

4.2 Generalized Approximation

The model derived by Küchemann predicts the behavior of the locus of aerodynamic centers for a wing of constant sweep. However, it is also of interest to obtain a more-general

model to be used for wings with piecewise-constant or non-constant sweep. Consider the wing with piecewise-constant sweep depicted in Fig. 4.2. For such a wing, Eq. (4.2) may be

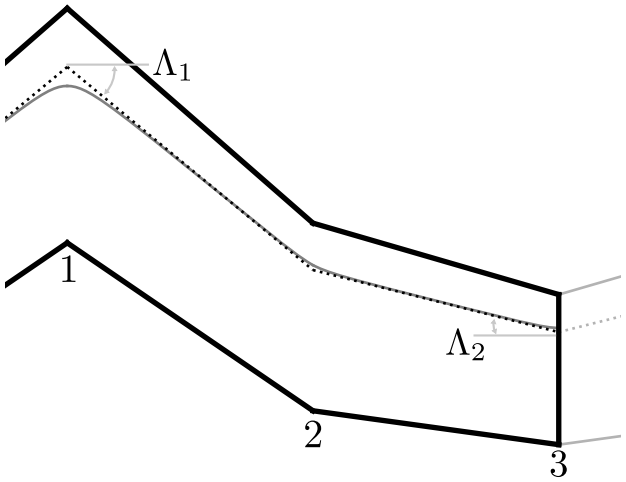

Fig. 4.2: The predicted locus of aerodynamic centers of a wing of large aspect ratio with piecewise-constant sweep (not to scale).

rewritten in the form

$$f(z) = f_{c/4}(z) + \frac{c(z)}{2\pi}\left(\lambda_1(\Lambda_1, z_1)\Lambda_1 + \lambda_2\Big(\frac{\Lambda_2 - \Lambda_1}{2}, z_2\Big)\frac{\Lambda_2 - \Lambda_1}{2} + \lambda_3(-\Lambda_2, z_3)(-\Lambda_2)\right) \quad (4.9)$$

where $f_{c/4}(z)$ is a function describing the quarter-chord line of the wing, z_1, z_2, and z_3 are distances along the z-axis from points 1, 2, and 3, respectively, and the interpolation functions, $\lambda_i(\Lambda_i, z_i)$, are of the same form as the interpolation function described in Eq. (4.3)

$$\lambda_i(\Lambda_i, z_i) = \sqrt{1 + \Big(2\pi\frac{\tan\Lambda_i}{\Lambda_i}\frac{z_i}{c(z_i)}\Big)^2} - 2\pi\frac{\tan\Lambda_i}{\Lambda_i}\frac{|z_i|}{c(z_i)} \quad (4.10)$$

Three hyperbolas are used in Eq. (4.9) to interpolate between the deviation of the aerodynamic center at points 1, 2, and 3, defined by Eq. (4.1), and the other points along the span.

Now, consider the wing depicted in Fig. 4.3, with the non-constant sweep distribution

$$\Lambda(z) = \frac{\Lambda_2 - \Lambda_1}{b/2} z + \Lambda_1 \tag{4.11}$$

For such a wing, the sweep angle is changing continuously along the span. For this wing,

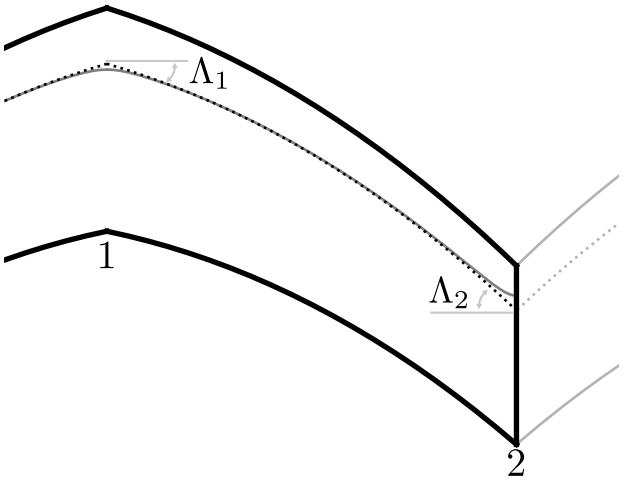

Fig. 4.3: The predicted locus of aerodynamic centers of a wing of large aspect ratio with non-constant sweep (not to scale).

Eq. (4.9) can be rewritten in the form

$$f(z) = f_{c/4}(z) + \frac{c(z)}{2\pi}\left(\lambda_1(\Lambda_1, z_1)\Lambda_1 + \int_0^{b/2} \lambda(\Lambda'(z_0), z - z_0)\Lambda'(z_0)dz_0 + \lambda_2(-\Lambda_2, z_2)(-\Lambda_2)\right) \tag{4.12}$$

where Λ' is the derivative of the sweep angle with respect to z. Notice that, like Eq. (4.9), Eq. (4.12) contains a term for the influence of the change in sweep along the span of the wing. However, because the sweep along the wing is constantly changing, the influence of the change in sweep along the wing becomes an integral. Even for simple geometries, such as that described by Eq. (4.11), the evaluation of the integral in Eq. (4.12) is a substantially involved process, and beyond the scope of this work. Therefore, the locus of aerodynamic centers shown in Fig. 4.3 was calculated without computing the integral, thus assuming the influence of the change in sweep along the wing is small.

The models described by Eqs. (4.9) and (4.12) assume wings of large aspect ratio, however, adjustments for aspect ratio, similar to those made by Küchemann, could be applied. For example, in the case of the piecewise-constant-sweep wing in Fig. 4.2, the sweep angles Λ_1 and Λ_2 could be adjusted with Eq. (4.4), using for the aspect ratio only the portion of the wing at each respective sweep angle. Similarly, there likely exist effective sweep angles and a K value for the non-constant-sweep wing in Fig. 4.3 that account for effects of the continuously changing sweep not captured in the large-aspect-ratio model described by Eq. (4.12).

Given the blind expansion of Küchemann's formulation to wings with piecewise-constant sweep and non-constant sweep, it is worth stating that the general models presented in Eqs. (4.9) and (4.12) are unvalidated, and are presented here solely for consideration in future work. All wings considered in this body of work are of constant sweep, therefore Eq. (4.5) is sufficient for use herein.

CHAPTER 5

PROPERTIES OF SWEPT WING SECTIONS

Recall from Eqs. (1.4) and (1.15) that, to predict the lift of a finite wing, lifting-line theory requires knowledge of the wing's section properties. In order to obtain that information, each wing section is modeled as an infinite wing with the same airfoil geometry as the corresponding wing section. Thin-airfoil theory is a traditional method of performing such analyses [3–5, 18, 46, 47]. In spite of the assumptions made in the theory, thin-airfoil theory provides valuable insight into the aerodynamic properties of airfoils (i.e. infinite wings). As such, it is discussed here before relaxing its assumptions to derive a more-general model for the section properties of infinite wings with sweep.

Assuming potential flow, infinite wings in thin-airfoil theory are modeled with a vortex distribution placed on the camber line of the wing. The strength of the vortex distribution is adjusted, perturbing the flow in such a way that the surface of the the wing becomes a stream surface of the flow, as depicted in Fig. 5.1 [1]. However, because the vortex sheet is located on the camber line, it is only able to create stream surfaces for thin airfoils with small amounts of camber, and only for small angles of attack. As the thickness, camber, or angle of attack increases, the vortex distribution is no longer able to produce a solution for the desired stream surface [46].

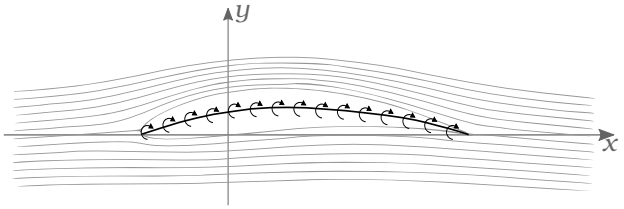

Fig. 5.1: Synthesis of an airfoil using a vortex distribution placed on the camber line of the airfoil section.

[1] Unless specified otherwise, all airfoils shown in this chapter are NACA 9515. This airfoil was chosen because its camber and thickness result in clear visualization of the ideas presented in the figures.

Under the assumptions of thin-airfoil theory, and following the development of Küchemmann [4], the change in pressure across the vortex sheet is proportional to the strength of the vortex distribution and the sine of the angle it makes with the freestream

$$\Delta P(x) = \rho V_\infty \gamma(x) \sin(\psi) \tag{5.1}$$

For a straight infinite wing, with no sideslip, the vortices and the freestream are normal to one another, and the pressure difference is

$$\Delta P(x) = \rho V_\infty \gamma(x) \tag{5.2}$$

An infinite wing with sweep is created from a straight infinite wing by translating the airfoil cross sections in their own respective plane, shearing the wing such that the new spanwise axis forms an angle Λ with the spanwise axis of the infinite straight wing, as shown in Fig. 5.2. The pressure difference for the infinite swept wing, with the relation $\psi = 90^o - \Lambda$, is thus described as

$$\Delta P_\Lambda(x) = \rho V_\infty \gamma(x) \cos(\Lambda) \tag{5.3}$$

Within the approximations of small angles and thin airfoils, this means that the section lift, per unit length, is found by integrating the pressure change along the airfoil, resulting in the expression

$$\tilde{L}_\Lambda = \rho V_\infty \Gamma \cos(\Lambda) \tag{5.4}$$

where Γ is the total circulation produced by the infinite wing per unit length along the z-axis. This lift is commonly non-dimensionalized by the dynamic pressure of the freestream, and the length of the chord, c, giving

$$\tilde{C}_{L_\Lambda} = \frac{\tilde{L}_\Lambda}{\frac{1}{2}\rho V_\infty^2 c} = \frac{2\Gamma}{V_\infty c} \cos \Lambda \tag{5.5}$$

Given the assumptions already made, it is assumed in thin-airfoil theory that Γ is not a function of the sweep angle. The ratio of section lift coefficients between the swept and

un-swept wings is thus predicted to be [4, 18, 47]

$$\frac{\tilde{C}_{L_\Lambda}}{\tilde{C}_L} = \cos(\Lambda) \tag{5.6}$$

where $\tilde{C}_{L_\Lambda}$ is the section lift coefficient of the swept wing and $\tilde{C}_L$ is the section lift coefficient of the un-swept wing. Thus, it is predicted that the section lift of an infinite wing decreases as the cosine of its sweep angle.

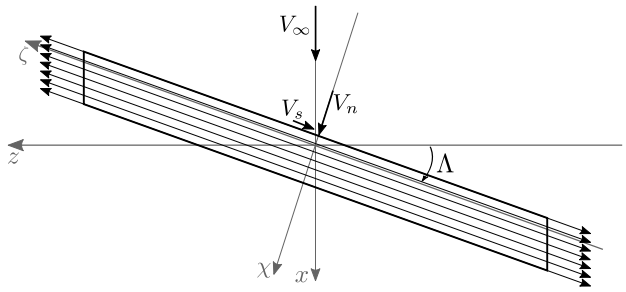

Fig. 5.2: Swept wing coordinate system, and depiction of the spanwise, V_s, and normal, V_n, freestream velocity components.

The approximation in Eq. (5.6) includes the assumptions of: a thin airfoil, small angles of attack, and no sideslip. It also assumes that the airfoil cross-section "seen" by the flow is the same as that of a straight wing. While this may be adequate for initial estimates, these assumptions can be relaxed to allow for a more general section-lift estimate that takes into account airfoil thickness and camber; larger angles of attack and sideslip; and changes in the effective airfoil cross-section.

It should be noted that the sweep angle, Λ, is considered positive for infinite wings sheared in the manner shown in Fig. 5.2, and negative if the shearing is reversed. This has no effect on the equations presented thus far, but is important once the idea of sideslip is introduced hereafter.

Within the approximations of thin-airfoil theory, the change in pressure across a thin vortex sheet described in Eq. (5.1) produces a moment about the leading edge of the sheet

described by the equation [18]

$$\Delta P_m(x) = -\rho V_\infty \gamma(x) x \sin(\psi) \tag{5.7}$$

where a positive moment is described by the right-hand rule about the negative z-axis. For the straight infinite wing, assuming no sideslip, the vortices and the freestream are normal to one another, and the moment caused by the pressure difference is

$$\Delta P_m(x) = -\rho V_\infty \gamma(x) x \tag{5.8}$$

Similarly, the moment caused by the pressure difference about the leading edge of the infinite swept wing, with the relation $\psi = 90^\circ - \Lambda$, is thus described as

$$\Delta P_{m_\Lambda}(x) = -\rho V_\infty \gamma(x) x \cos^2(\Lambda) \tag{5.9}$$

The section moment about the leading edge of the wing, per unit length, is found by integrating the pressure change along the airfoil, resulting in the expression

$$\tilde{m}_\Lambda = -\rho V_\infty \cos^2(\Lambda) \int_0^c \gamma(x) x dx \tag{5.10}$$

Non-dimensionalizing by the dynamic pressure of the freestream and the length of the chord squared, gives

$$\tilde{C}_{m_\Lambda} = \frac{\tilde{m}_\Lambda}{\frac{1}{2}\rho V_\infty^2 c^2} = \frac{-2\cos^2(\Lambda)}{V_\infty c^2} \int_0^c \gamma(x) x dx \tag{5.11}$$

Given the assumptions already made, it is assumed that $\gamma(x)$ is not a function of the sweep angle. The ratio of section lift coefficients between the swept and un-swept wings is thus predicted to be

$$\frac{\tilde{C}_{m_\Lambda}}{\tilde{C}_m} = \cos^2(\Lambda) \tag{5.12}$$

where $\tilde{C}_{m_\Lambda}$ is the section moment coefficient of the swept wing and $\tilde{C}_m$ is the section moment coefficient of the un-swept wing. Thus, in thin-airfoil theory it is predicted that the section

moment about the leading edge of an infinite wing decreases as the square of the cosine of its sweep angle.

Therefore, using thin-airfoil theory, predictions for a section's lift and moment are given by the simple models in Eqs. (5.6) and (5.12). It is shown in this chapter that these models can be improved if the assumptions made in thin-airfoil theory are relaxed.

5.1 Influence of Sweep on the Effective Freestream Velocity

Consider a wing of infinite length and a constant sweep angle, Λ. When the sweep angle is zero, the wing can be described by the x-y-z coordinate system shown in Figs. 5.1 and 5.2. For a wing with non-zero sweep, it can be more convenient to define a second χ-y-ζ coordinate system that is a right-hand rotation of the x-y-z coordinate system about the negative y-axis by the angle Λ, as seen in Fig. 5.2.

Recall that potential flow models, like thin-airfoil theory, use elements (e.g. sources/sinks, vortices, uniform flow, etc.) to force a streamline of the flow to lie on the surface of the wing being modeled. For a wing with sweep, the freestream velocity can be decomposed into a spanwise component, aligned with the ζ-axis, and a normal component, in the χ-y plane. The spanwise component of the flow is at all points tangential to the wing surface and parallel to the wing's circulation. Therefore, it does not affect the calculation of surface streamlines. As such, only the normal component of the freestream contributes to the predicted lift of the wing [46].

To find expressions for the component of the freestream that contributes to the lift, the freestream velocity, as a function of angle of attack, α, is first described in the x-y-z coordinate system as

$$\vec{V}_\infty = V_\infty \cos\alpha \, \hat{\imath}_x + V_\infty \sin\alpha \, \hat{\imath}_y \tag{5.13}$$

The component of this velocity that flows parallel to the ζ-axis, $\vec{V}_s$, is found by projecting $\vec{V}_\infty$ onto the ζ-axis. Once the spanwise component is known, the component of the freestream in the χ-y plane, $\vec{V}_n$, is described by subtracting $\vec{V}_s$ from $\vec{V}_\infty$. In the x-y-z coordinate system,

the resulting expressions are

$$\vec{V}_s = V_\infty \cos\alpha \sin\Lambda(\sin\Lambda\,\hat{\imath}_x - \cos\Lambda\,\hat{\imath}_z) \tag{5.14}$$

$$\vec{V}_n = V_\infty \cos\alpha \cos^2\Lambda\,\hat{\imath}_x + V_\infty \sin\alpha\,\hat{\imath}_y + V_\infty \cos\alpha \sin\Lambda\cos\Lambda\,\hat{\imath}_z \tag{5.15}$$

The spanwise and normal velocities are more simply expressed by rewriting them in the χ-y-ζ coordinate frame as follows

$$\vec{V}_s = -V_\infty \cos\alpha \sin\Lambda\,\hat{\imath}_\zeta \tag{5.16}$$

$$\vec{V}_n = V_\infty \cos\alpha \cos\Lambda\,\hat{\imath}_\chi + V_\infty \sin\alpha\,\hat{\imath}_y \tag{5.17}$$

with magnitudes

$$V_s = V_\infty \cos\alpha \sin\Lambda \tag{5.18}$$

$$V_n = V_\infty \sqrt{\cos^2\alpha\cos^2\Lambda + \sin^2\alpha} \tag{5.19}$$

Because $\vec{V}_n$ is the component of the freestream that affects the lift on the wing, it shall heretofore be referred to as the *effective freestream velocity*. Figure 5.3 shows the variation of the ratio of V_n to V_∞ as a function of sweep, for four angles of attack. At zero degrees angle of attack, the effective freestream maintains more than 90% of the freestream velocity's magnitude up to a sweep angle of almost 25°, before decreasing to 50% of the freestream at $\Lambda = 60°$. As angle of attack increases, the effect of sweep on the effective freestream is slightly lessened.

5.2 Influence of Sweep on the Effective Angle of Attack

The freestream angle of attack is defined as the angle the freestream makes with the plane of the wing (the x-z plane). Assuming zero sideslip, the angle of attack can be written as

$$\alpha = \tan^{-1}\left(\frac{V_y}{V_x}\right) \tag{5.20}$$

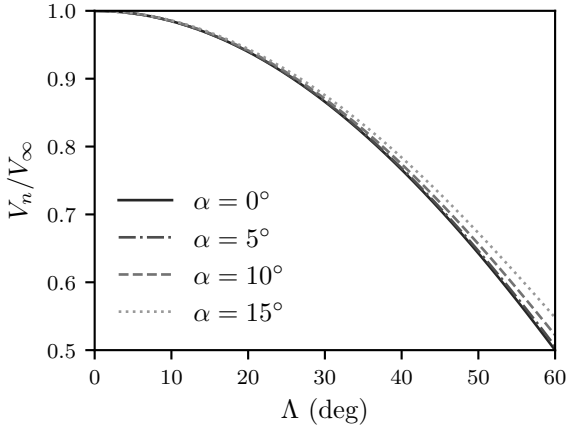

Fig. 5.3: The change in effective freestream velocity, V_n, with respect to sweep.

where V_x and V_y are the x and y components of the freestream from Eq. (5.13). The process for determining the effective freestream velocity, $\vec{V}_n$, reveals a change in the effective value for the angle of attack. The y component of $\vec{V}_\infty$ is equal to the y component of $\vec{V}_n$, whereas the χ component of $\vec{V}_n$ is less than the x component of $\vec{V}_\infty$, creating an effective change in angle of attack, as can be seen in Fig. 5.4.

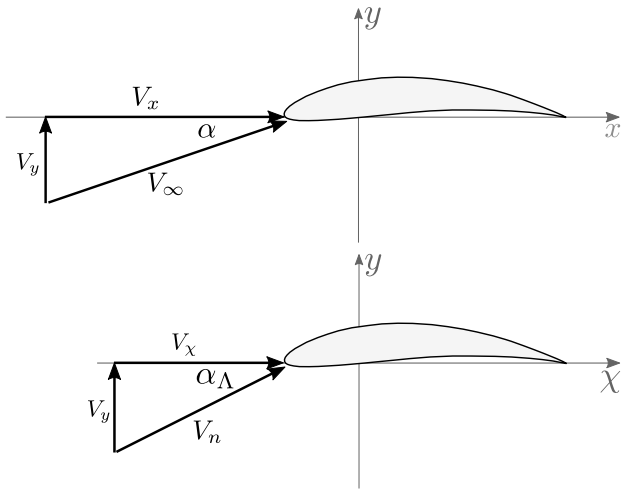

Fig. 5.4: The effect of sweep on the effective angle of attack, α_Λ.

To find the angle the effective freestream makes with the χ-ζ plane, Eq. (5.17) is used in Eq. (5.20), resulting in the expression

$$\alpha_\Lambda = \tan^{-1}\left(\frac{V_y}{V_\chi}\right) = \tan^{-1}\left(\frac{V_\infty \sin\alpha}{V_\infty \cos\alpha \cos\Lambda}\right) = \tan^{-1}\left(\frac{\tan\alpha}{\cos\Lambda}\right) \tag{5.21}$$

The ratio of effective angle of attack, α_Λ, to the freestream angle of attack, α, as a function of the sweep angle, is depicted in Fig. 5.5, at multiple angles of attack. The effective angle of attack is about 6% higher than α at $\Lambda = 20°$, but is almost double at 60° sweep. In the case that the angle of attack is exactly zero, however, the effective angle of attack remains zero for all sweep angles. As the angle of attack increases, the effect that the sweep angle has on the effective angle of attack is slightly diminished.

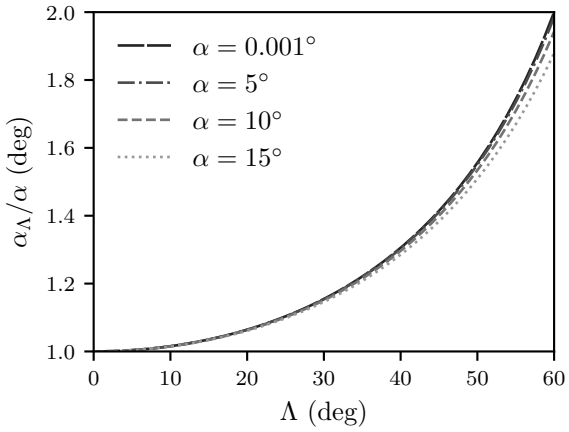

Fig. 5.5: The change in effective angle of attack, α_Λ, with respect to sweep.

5.3 Generalization of the Influence of Sweep to Include Sideslip

The effective freestream, $\vec{V}_n$, and effective angle of attack, α_Λ, can be further generalized by allowing the freestream velocity vector to deviate from the x-y plane by the sideslip angle, β, defined as

$$\beta = \tan^{-1}\left(\frac{V_z}{V_x}\right) \tag{5.22}$$

Including sideslip, the freestream vector from Eq. (5.13) becomes

$$\vec{V}_\infty = V_\infty \frac{\cos\alpha\cos\beta}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_x + V_\infty \frac{\sin\alpha\cos\beta}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_y + V_\infty \frac{\cos\alpha\sin\beta}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_z \quad (5.23)$$

Following the procedure used previously, the spanwise and normal velocity components are found to be

$$\vec{V}_s = V_\infty \frac{\cos\alpha\sin(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}(\sin\Lambda\,\hat{\imath}_x - \cos\Lambda\,\hat{\imath}_z) \quad (5.24)$$

$$\vec{V}_n = V_\infty \frac{\cos\alpha\cos\Lambda\cos(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_x + V_\infty \frac{\sin\alpha\cos\beta}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_y + V_\infty \frac{\cos\alpha\sin\Lambda\cos(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_z \quad (5.25)$$

Describing these two components in the χ-y-ζ coordinate system yields

$$\vec{V}_s = -V_\infty \frac{\cos\alpha\sin(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_\zeta \quad (5.26)$$

$$\vec{V}_n = V_\infty \frac{\cos\alpha\cos(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_\chi + V_\infty \frac{\sin\alpha\cos\beta}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\,\hat{\imath}_y \quad (5.27)$$

with magnitudes

$$V_s = V_\infty \frac{\cos\alpha\sin(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}} \quad (5.28)$$

$$V_n = V_\infty \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}} \quad (5.29)$$

Using Eq. (5.27) in Eq. (5.20), the effective angle of attack is found to be

$$\alpha_\Lambda = \tan^{-1}\left(\frac{V_y}{V_\chi}\right) = \tan^{-1}\left(\frac{V_\infty\sin\alpha\cos\beta}{V_\infty\cos\alpha\cos(\Lambda-\beta)}\right) = \tan^{-1}\left(\frac{\tan\alpha\cos\beta}{\cos(\Lambda-\beta)}\right) \quad (5.30)$$

Finally, an effective sideslip is defined, using Eqs. (5.26) and (5.27) in Eq. (5.22), to provide a measure of the component of the spanwise flow

$$\beta_\Lambda = \beta - \Lambda \quad (5.31)$$

Note that, here, the sign of the sweep angle is important.

5.4 Influence of Sweep on the Effective Airfoil

Because only the normal component of the velocity is relevant when determining the section lift of an infinite wing with sweep, the effective airfoil geometry influencing the flow is the cross-section of the wing in the χ-y plane—the same plane as the effective freestream velocity (Fig. 5.2) [47]. This new, effective airfoil maintains the y coordinates defining the surface of the original airfoil, however, the x locations of the surface are projected from the x-y plane to the χ-y plane, resulting in the scaling

$$\chi = x \cdot \cos \Lambda \tag{5.32}$$

Because the airfoil scaling occurs only along one axis, the geometric–and thus aerodynamic–properties of the airfoil also change, as depicted in Fig. 5.6.

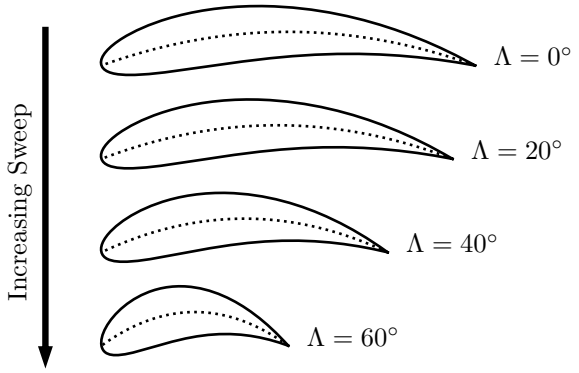

Fig. 5.6: The change in effective airfoil geometry with sweep.

As the amount of sweep increases, the relative maximum camber and maximum thickness of the airfoil increase. The relative location of the maximum camber, on the other hand, remains constant. For the airfoil shown in Fig. 5.6, the relative maximum thickness and camber increase by just over 6% for $\Lambda = 20°$, about 30% for $\Lambda = 40°$, and 100% for $\Lambda = 60°$.

5.5 Influence of Sweep on the Section-Lift Coefficient

Consider a finite section of an infinite wing, defined by the differential length dl along the z-axis. When determining the lift of this finite section for an infinite wing with sweep, it is necessary to use the effective freestream velocity and its corresponding effective angle of attack, as well as the effective airfoil geometry. From Eq. (5.1), the section lift found using the circulation strength of the effective airfoil, Γ_Λ, and the normal velocity, V_n, described by Eq. (5.29) is

$$dL_\Lambda = \rho V_n \Gamma_\Lambda dl_\Lambda = \rho V_\infty \Gamma_\Lambda \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{dl}{\cos\Lambda} \tag{5.33}$$

where dl_Λ is the differential length along the ζ-axis, shown in Fig. 5.7. The section-lift

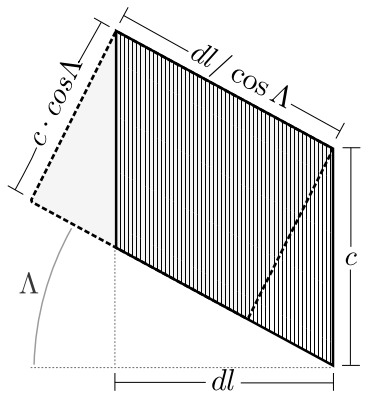

Fig. 5.7: Comparison of the area of a spanwise section of a swept wing using the original airfoil section (shaded) and the effective airfoil section (not shaded).

coefficient is found by normalizing dL_Λ by the dynamic pressure of the freestream, V_∞, the differential length, dl, and the chord, c. The resulting section lift coefficient is given by

$$\tilde{C}_{L_\Lambda} = \frac{dL_\Lambda}{\frac{1}{2}\rho V_\infty^2 c\, dl} = \frac{2V_n\Gamma_\Lambda}{V_\infty^2 c\cos\Lambda} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\cos\Lambda\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{2\Gamma_\Lambda}{V_\infty c} \tag{5.34}$$

It can be seen in Eq. (5.34) that the section lift coefficient of an infinite wing with sweep maintains the same base form as that for a straight wing, shown in Eq. (5.5). The key differences are the trigonometric coefficient that accounts for the change in effective

freestream and Γ_Λ, which accounts for the change in effective airfoil circulation and effective angle of attack. Equation (5.34) can be rewritten in the form

$$\frac{\tilde{C}_{L_\Lambda}}{\tilde{C}_L} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\cos\Lambda\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{\Gamma_\Lambda}{\Gamma} \tag{5.35}$$

If there exists a definable relationship between Γ and Γ_Λ, Eq. (5.35) provides a prediction of the section-lift coefficient of an infinite wing with sweep, based solely on the known properties of the corresponding un-swept wing. This section lift prediction can then be used in aerodynamic models to increase their accuracy for finite swept wings [14, 32].

5.5.1 Conformal Mapping

A first effort to find the relationship Γ_Λ/Γ can be made using complex potential flow. From the theory of conformal mapping [18, 48], flow over a lifting cylinder can be mapped to flow over an arbitrary airfoil with the mapping

$$w(\omega) = \omega + \sum_{n=1}^{\infty}\frac{C_n}{\omega^n} \tag{5.36}$$

where ω is in the plane of the lifting cylinder and C_n are complex constants. Using this mapping, the circulation produced by the airfoil is [18, 48]

$$\Gamma = 4\pi V_\infty R\sin(\alpha-\alpha_{L0}) \tag{5.37}$$

where R is the radius of the lifting cylinder and α_{L0} is the zero-lift angle of attack. In order to relate this Γ to Γ_Λ, a second mapping is needed to scale the real(w) axis, as a function of Λ

$$w' = \text{real}(w)\cos\Lambda + i\,\text{imag}(w) = w\cdot\frac{\cos\Lambda+\tan^2(\arg(w))+i\,\tan(\arg(w))(1-\cos\Lambda)}{1+\tan^2(\arg(w))} \tag{5.38}$$

However, this additional factor is not complex differentiable, and thus is not a conformal mapping. This is quickly seen by examining the Cauchy-Riemann equations [26]. If a

mapping is complex differentiable, the following must be true

$$w = u(x,y) + i\,v(x,y)$$

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \qquad (5.39)$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}$$

However, for the proposed mapping in Eq. (5.38)

$$w' = x\cos(\Lambda) + i\,y$$

$$\frac{\partial u}{\partial x} = \cos(\Lambda) \neq 1 = \frac{\partial v}{\partial y} \qquad (5.40)$$

$$\frac{\partial v}{\partial x} = 0 = -\frac{\partial u}{\partial y}$$

So, an other means must be used to find Γ_Λ/Γ.

5.5.2 Curve Fits

The next option for relating the circulations of the original and effective airfoil sections is to use empirical relationships. In order to identify such relationships, it is convenient to rewrite the circulation produced by an effective airfoil, Γ_Λ, as the following function of the effective angle of attack, α_Λ, and the effective sideslip angle, β_Λ

$$\Gamma_\Lambda = \frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}}\Gamma_{\Lambda,\alpha_\Lambda}\left(\alpha_\Lambda - \alpha_{L0_\Lambda}\right) \qquad (5.41)$$

where $\Gamma_{\Lambda,\alpha_\Lambda}$ is the change in effective section circulation with respect to the effective angle of attack, and α_{L0_Λ} is the effective angle of attack at which no lift is generated.

In Eq. (5.41), the product of the effective circulation slope, $\Gamma_{\Lambda,\alpha_\Lambda}$, and the difference between the effective of attack and the effective zero-lift angle of attack form a linear model for the effective section circulation as a function of the effective angle of attack. In practice,

the parameters in this linear model, $\Gamma_{\Lambda,\alpha_\Lambda}$ and α_{L0_Λ}, are found by calculating the circulation produced by the effective airfoil geometry in a flow with magnitude V_∞ at various angles of attack and zero effective sideslip. A line is then fit to the results to predict $\Gamma_{\Lambda,\alpha_\Lambda}$ and α_{L0_Λ}. This methodology for approximating the effective circulation, Γ_Λ, results in values for $\Gamma_{\Lambda,\alpha_\Lambda}$ and α_{L0_Λ} that are constants for a given effective airfoil geometry, independent of the flow conditions. However, the circulation produced by the effective airfoil geometry is not solely a linear function of the effective angle of attack, but is also a function of the effective sideslip, β_Λ. As seen from Eqs. (5.29)–(5.31), for $\beta_\Lambda \neq 0$, changes in the effective angle of attack correspond to changes in the effective freestream velocity, effectively reducing the change in the airfoil's circulation for a given change in effective angle of attack. The trigonometric term in Eq. (5.41), derived from Eq. (5.29) for a wing with no sweep and non-zero sideslip, accounts for that reduction.

Non-dimensionalizing the effective circulation slope, $\Gamma_{\Lambda,\alpha_\Lambda}$, results in the effective lift slope

$$\tilde{C}_{L_{\Lambda,\alpha_\Lambda}} = \frac{2\Gamma_{\Lambda,\alpha_\Lambda}}{V_\infty c \cos\Lambda} \tag{5.42}$$

Thus, using Eqs. (5.41) and (5.42), the section lift coefficient for a wing section, defined in Eq. (5.34), can be rewritten as

$$\tilde{C}_{L_\Lambda} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}} \frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}} \tilde{C}_{L_{\Lambda,\alpha_\Lambda}}\left(\alpha_\Lambda - \alpha_{L0_\Lambda}\right) \tag{5.43}$$

The ratio of the effective airfoil's lift coefficient to the un-swept airfoil's lift coefficient, from Eq. (5.35), is then

$$\frac{\tilde{C}_{L_\Lambda}}{\tilde{C}_L} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}} \frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}} \frac{\tilde{C}_{L_{\Lambda,\alpha_\Lambda}}}{\tilde{C}_{L,\alpha}} \frac{(\alpha_\Lambda - \alpha_{L0_\Lambda})}{(\alpha - \alpha_{L0})} \tag{5.44}$$

This expression may be displayed in terms of the the ratio of the effective airfoil's lift slope to the un-swept airfoil's lift slope

$$R_{\tilde{C}_{L,\alpha}} = \frac{\tilde{C}_{L_\Lambda,\alpha_\Lambda}}{\tilde{C}_{L,\alpha}} \tag{5.45}$$

and the difference in the zero-lift angle of attack of the effective and un-swept airfoils,

$$\Delta\alpha_{L0} = \alpha_{L0_\Lambda} - \alpha_{L0} \tag{5.46}$$

in radians.

The lift coefficient scaling factor described in Eq. (5.44) has two parts: the scaling that comes from a change in geometry relative to the freestream and the scaling that comes from the change in the airfoil's effective aerodynamic properties. In this derivation, the former is found using analytical formulations, without assumption of airfoil shape, whereas the latter must be found using empirical relations obtained later in this chapter.

5.6 Influence of Sweep on the Section-Moment Coefficient

Consider the description of the moment produced by a finite section of an infinite wing with sweep, given by

$$dm_\Lambda = -\rho V_n \Gamma_{m_\Lambda} dl_\Lambda = \rho V_\infty \Gamma_{m_\Lambda} \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}} \frac{dl}{\cos\Lambda} \tag{5.47}$$

where

$$\Gamma_{m_\Lambda} = \oint \vec{r}(s) \times \gamma_\Lambda(s)\hat{n}(s)\, ds \tag{5.48}$$

and dl_Λ is the differential length along the ζ-axis, shown in Fig. 5.7, s is a location along the surface of the swept airfoil, $\vec{r}$ is a vector from the leading edge of the airfoil to the surface of the airfoil, and $\hat{n}$ is a unit vector normal to the surface of the airfoil.

The section-moment coefficient is found by non-dimensionalizing dm_Λ by the dynamic pressure of the freestream, V_∞, the differential length, dl, and the chord squared, c^2. The

resulting section moment coefficient is given by

$$\tilde{C}_{m_\Lambda} = \frac{dm_\Lambda}{\frac{1}{2}\rho V_\infty^2 c^2 dl} = \frac{2V_n \Gamma_{m_\Lambda}}{V_\infty^2 c^2 \cos\Lambda} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\cos\Lambda\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{2\Gamma_{m_\Lambda}}{V_\infty c^2} \tag{5.49}$$

Equation (5.49) can be rewritten as the ratio of the moment coefficient for a corresponding un-swept wing section

$$\frac{\tilde{C}_{m_\Lambda}}{\tilde{C}_m} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\cos\Lambda\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{\Gamma_{m_\Lambda}}{\Gamma_m} \tag{5.50}$$

As was done with the lift, the relationship between Γ_m and Γ_{m_Λ} is found by fitting data to curves. In that way, Eq. (5.49) provides a prediction of the section-moment coefficient of an infinite wing with sweep, based solely on the known properties of the corresponding un-swept wing.

5.6.1 Curve Fits

Following the process used with Eq. (5.41), it is convenient to rewrite the circulation moment produced by an effective airfoil, Γ_{m_Λ}, as the following function of the effective angle of attack, α_Λ, and the effective sideslip angle, β_Λ

$$\Gamma_{m_\Lambda} = \frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}}\Gamma_{m_{\Lambda,\alpha_\Lambda}}\left(\alpha_\Lambda - \alpha_{m0_\Lambda}\right) \tag{5.51}$$

where $\Gamma_{m_{\Lambda,\alpha_\Lambda}}$ is the change in effective section circulation moment with respect to the effective angle of attack. Non-dimensionalizing $\Gamma_{m_{\Lambda,\alpha_\Lambda}}$ results in the effective moment slope

$$\tilde{C}_{m_{\Lambda,\alpha_\Lambda}} = \frac{2\Gamma_{m_{\Lambda,\alpha_\Lambda}}}{V_\infty c^2\cos\Lambda} \tag{5.52}$$

Thus, using Eqs. (5.51) and (5.52), the section moment coefficient for a wing section, defined in Eq. (5.49), can be rewritten as

$$\tilde{C}_{m_\Lambda} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}}\tilde{C}_{m_\Lambda,\alpha_\Lambda}\left(\alpha_\Lambda-\alpha_{m0_\Lambda}\right) \tag{5.53}$$

The ratio of the effective airfoil's moment coefficient to the un-swept airfoil's lift coefficient, from Eq. (5.50), is then

$$\frac{\tilde{C}_{m_\Lambda}}{\tilde{C}_m} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}}\frac{\tilde{C}_{m_\Lambda,\alpha_\Lambda}}{\tilde{C}_{m,\alpha}}\frac{(\alpha_\Lambda-\alpha_{m0_\Lambda})}{(\alpha-\alpha_{m0})} \tag{5.54}$$

This expression may be displayed in terms of the the ratio of the effective airfoil's moment slope to the un-swept airfoil's moment slope

$$R_{\tilde{C}_{m,\alpha}} = \frac{\tilde{C}_{m_\Lambda,\alpha_\Lambda}}{\tilde{C}_{m,\alpha}} \tag{5.55}$$

and the difference in the zero-moment angle of attack of the effective and un-swept airfoils

$$\Delta\alpha_{m0} = \alpha_{m0_\Lambda} - \alpha_{m0} \tag{5.56}$$

in radians.

5.7 Influence of Sweep on the Section-Drag Coefficient

Consider the drag over a finite section of an infinite wing, defined by the finite length dl along the z-axis. Because the wing under consideration is infinite, there must not exist any spanwise effects on the drag (e.g. spanwise boundary layer growth). Therefore, it is assumed that the drag experienced by the finite section is only a function of the component of the flow normal to the wing, as are the lift and moment. This conjecture does not hold for physical wings, but is valid for the discussion of infinite wings. Once the effective drag of the section is determined, the effective drag coefficient of this finite section of wing can

thus be written

$$\tilde{C}_{D_\Lambda} = \frac{\tilde{D}_\Lambda}{\frac{1}{2}\rho V_\infty^2 c \cos\Lambda} \tag{5.57}$$

where $\tilde{D}_\Lambda$ is calculated using the effective freestream magnitude, effective angle of attack, and effective airfoil geometry. The cosine of the sweep angle in the denominator of Eq. (5.57) results from the difference between a differential length along the z-axis and a differential length along the ζ-axis (shown in Fig. 5.7).

The effective drag coefficient described in Eq. (5.57) describes the force parallel to the effective freestream described by Eq. (5.27). Drag is defined as the force acting parallel to the freestream. Therefore, the effective drag coefficient from Eq. (5.57) can be decomposed into a component parallel to the freestream, and a component referred to as the side-force

$$\tilde{C}_{D_{\Lambda\infty}} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\tilde{C}_{D_\Lambda} \tag{5.58}$$

$$\tilde{C}_{S_\Lambda} = \frac{\cos\alpha\sin(\Lambda-\beta)}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\tilde{C}_{D_\Lambda} \tag{5.59}$$

Unlike the lift and moment coefficients, the drag of an infinite wing is not an explicit function of the circulation of the wing [3, 18]. Because of the more-complex nature of drag, the drag coefficient can not be modeled in the same manner as the lift and moment coefficients in the previous sections, but is modeled only in the general sense described by Eqs. (5.57), (5.58), and (5.59).

5.8 Application of Swept Wing Section Properties to a Vortex Panel Method

A two-dimensional vortex panel method is a common method used to model flow over an infinite straight wing. The heuristic of this method will be discussed here, but the details can be found in standard aerodynamics textbooks [3, 5, 17, 18].

An infinite wing is represented in a two-dimensional vortex panel method by a vortex sheet wrapped around the surface of the wing and discretized into a finite number of vortex panels, as shown in Fig. 5.8. The strength of each vortex panel is found numerically such that a streamline of the flow falls along the surface of the wing being modeled. This is very

similar to the process employed by thin-airfoil theory, as seen in Fig. 5.1, but is able to account for airfoil geometries with non-negligible thickness and camber, and higher angles of attack.

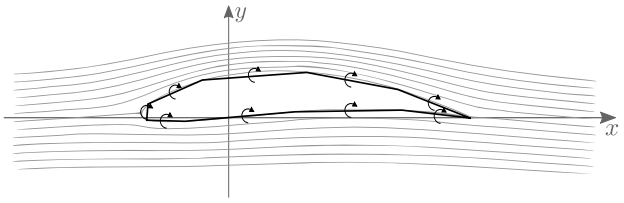

Fig. 5.8: Synthesis of an airfoil using a number of vortex panels placed on the surface of the airfoil section.

Results are obtained from the vortex panel method, for the case of an infinite wing with sweep, by modifying the freestream velocity magnitude, angle of attack, and airfoil geometry according to the discussion in the previous sections—Eqs. (5.29), (5.30), and (5.32), respectively. The total circulation predicted by the method is used in Eq. (5.34) to give the section-lift coefficient. Source code applying such a vortex panel method is given in Appendix B.1.

5.8.1 Vortex Panel Method Data, Lift Coefficient

Using the vortex panel method, circulation values are calculated for a range of NACA 4-digit airfoils and their effective airfoils for a range of sweep angles and angles of attack. For each airfoil at each sweep angle, $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$ and α_{L0_Λ} are found by fitting Eq. (5.41) to the circulation calculated over a range of effective angles of attack. The ratios of section lift slope and differences in zero-lift angle of attack for this range of airfoils are shown in Figs. 5.9 and 5.10. For the results included herein, all cases use an un-swept chord length and freestream velocity of unity.

5.8.1.1 Ratio of Circulation Slopes

From the vortex panel method results displayed in Fig. 5.9, it can be seen that $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$ can be approximated as a function only of the airfoil thickness. The data are fit to curves

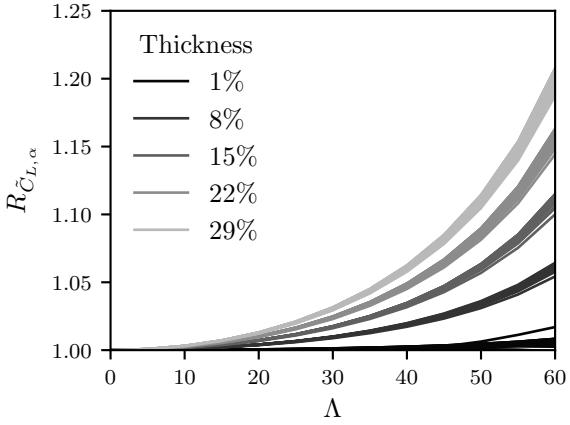

Fig. 5.9: The ratio of section lift slopes, $R_{\tilde{C}_{L,\alpha}}$, as a function of sweep. Each curve represents a NACA 4-digit airfoil.

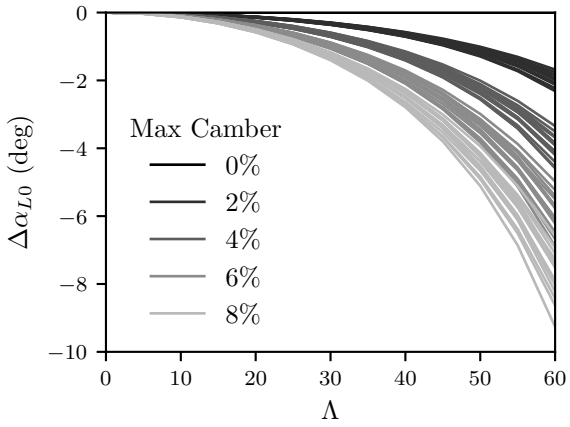

Fig. 5.10: The difference in zero-lift angle of attack, $\Delta\alpha_{L0}$, as a function of sweep. Each curve represents a NACA 4-digit airfoil.

of the form

$$R_{\tilde{C}_{L,\alpha}}(\tau) = \frac{1}{1 - f(\tau)\,\Lambda^2 - g(\tau)\,\Lambda^4} \tag{5.60}$$

where τ is the maximum thickness of the unswept airfoil expressed as a fraction of the unswept chord, Λ is the sweep expressed in radians, and $f(\tau)$ and $g(\tau)$ are exponential functions of the thickness. Performing a least-squares regression on the vortex panel method

data, collected using NACA 4-digit airfoils, results in the model

$$R_{\tilde{C}_{L,\alpha}}(\tau) = \frac{1}{1 - 0.2955\tau^{0.96}\,\Lambda^2 - 0.1335\tau^{0.68}\,\Lambda^4} \tag{5.61}$$

The mean error in using Eq. (5.61), for the results shown in Fig. 5.9, ranges from 0% at $\Lambda = 0°$, to a maximum of $0.46\% \pm 0.31\%$ at $\Lambda = 60°$.

5.8.1.2 Difference in Zero-Lift Angle of Attack

Based on the apparent trends of $\Delta\alpha_{L0}$, seen in Fig. 5.10, a curve of the form

$$\Delta\alpha_{L0}(\kappa) = \frac{1}{1 + h(\kappa)\,\Lambda^2 + i(\kappa)\,\Lambda^4} - 1 \tag{5.62}$$

is used to fit the data—where κ is the maximum camber, expressed as a fraction of the unswept airfoil's chord. Expressing $h(\kappa)$ and $i(\kappa)$ as exponential functions of the maximum camber, κ, Eq. (5.62) becomes

$$\Delta\alpha_{L0}(\kappa) = \frac{1}{1 + 0.5824\kappa^{0.92}\,\Lambda^2 + 1.3892\kappa^{1.16}\,\Lambda^4} - 1 \tag{5.63}$$

in radians. The mean error in using Eq. (5.63), for the results shown in Fig. 5.10, ranges from $0°$ at $\Lambda = 0°$, to a maximum of $0.28° \pm 0.33°$ at $\Lambda = 60°$.

5.8.2 Vortex Panel Method Data, Moment Coefficient

In a similar manner as with the lift coefficient, the vortex panel method is used to predict $\tilde{C}_{m_\Lambda,\alpha_\Lambda}$ and α_{m0_Λ} by fitting Eq. (5.51) to the computed circulation values. The ratios of section moment slope and differences in zero-moment angle of attack, for a range of NACA 4-digit airfoils, are shown in Figs. 5.11 and 5.12.

5.8.2.1 Ratio of Circulation Moment Slopes

From the vortex panel method results displayed in Fig. 5.11, it can be seen that the change in $\tilde{C}_{m_\Lambda,\alpha_\Lambda}$ can be approximated as a function only of the airfoil thickness, though

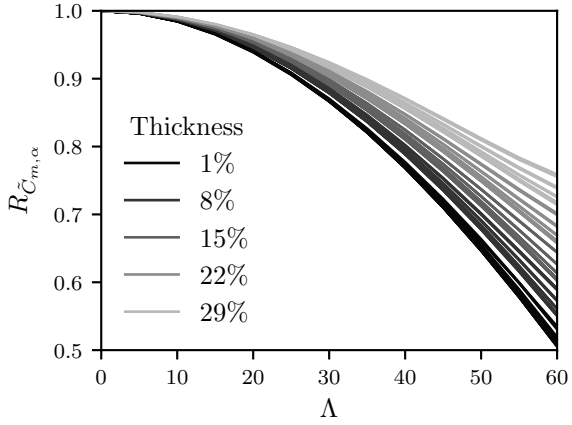

Fig. 5.11: The ratio of section moment slopes, $R_{\tilde{C}_{m,\alpha}}$, as a function of sweep. Each curve represents a NACA 4-digit airfoil.

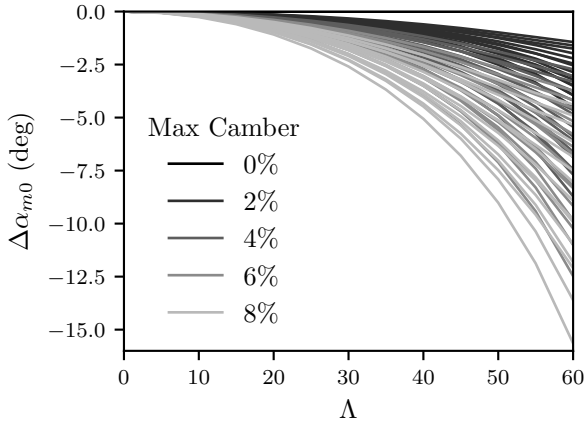

Fig. 5.12: The difference in zero-moment angle of attack, $\Delta\alpha_{m0}$, as a function of sweep. Each curve represents a NACA 4-digit airfoil.

there is also an effect from the camber. The data are fit to curves of the form

$$R_{\tilde{C}_{m,\alpha}}(\tau) = 1 + f(\tau)\Big(\cos\big(g(\tau)\,\Lambda\big) - 1\Big) \tag{5.64}$$

where τ is the maximum thickness of the unswept airfoil expressed as a fraction of the unswept chord, Λ is the sweep expressed in radians, and $f(\tau)$ and $g(\tau)$ are polynomial functions of the thickness. Performing a least-squares regression on the vortex panel method

data results in the model

$$R_{\tilde{C}_{m,\alpha}}(\tau) = 1 + (-2.37\tau + 0.91)\Big(\cos\left((6.62\tau^2 + 1.06)\,\Lambda\right) - 1\Big) \tag{5.65}$$

The mean error in using Eq. (5.65), for the results shown in Fig. 5.11, ranges from 0% at $\Lambda = 0°$, to a maximum of 2.25% $\pm$ 1.76% at $\Lambda = 60°$.

5.8.2.2 Difference in Zero-Moment Angle of Attack

The trends of $\Delta\alpha_{m0}$, seen in Fig. 5.12, suggest that the difference in zero-moment angle of attack is a function of the thickness and camber of the airfoil. Nevertheless, a curve of the form

$$\Delta\alpha_{m0}(\kappa) = \frac{1}{1 + h(\kappa)\,\Lambda^2 + i(\kappa)\,\Lambda^4} - 1 \tag{5.66}$$

is used to fit the data—where κ is the maximum camber, expressed as a fraction of the unswept airfoil's chord. Expressing $h(\kappa)$ and $i(\kappa)$ as exponential functions of the maximum camber, κ, Eq. (5.66) becomes

$$\Delta\alpha_{m0}(\kappa) = \frac{1}{1 + 1.07\kappa^{0.95}\,\Lambda^2 + 0.56\kappa^{0.83}\,\Lambda^4} - 1 \tag{5.67}$$

in radians. The mean error in using Eq. (5.67), for the results shown in Fig. 5.12, ranges from 0% at $\Lambda = 0°$, to a maximum of $1.45° \pm 1.45°$ at $\Lambda = 60°$, despite the omission of thickness in Eq. (5.66).

5.8.3 Vortex Panel Method Data, Drag Coefficient

The analyses included thus far have all assumed inviscid flow, however, the formulations for the effective freestream velocity magnitude, angle of attack, and airfoil geometry for infinite wings with sweep may also be applied to more complicated aerodynamic property prediction algorithms, such as those used in XFOIL [49]. XFOIL uses boundary layer integration techniques to predict the effects of viscosity on section data. Using XFOIL, a prediction for the change in section drag coefficient may also be developed, though not

accounting for the boundary layer growth that occurs in the spanwise direction on swept wings. The drag coefficients obtained are scaled to describe the section drag per unit length along the z-axis

$$\tilde{C}_{D_\Lambda} = \frac{\tilde{C}_{D_{\text{XFOIL}}}}{\cos\Lambda} \tag{5.68}$$

This effective drag coefficient describes the force parallel to the effective freestream, and can be decomposed into a drag component parallel to the freestream, using Eq. (5.58), and a side-force component, using Eq. (5.59).

Section drag data were gathered for eight airfoils in the NACA 4-digit family for a range of sweep angles and angles of attack. The values of the drag coefficient were found using the parameters: ncrit = 9.0, xtr = 0.1, and Re = 1×10^6. The results are shown in Fig. 5.13.

5.9 Validation of the Vortex Panel Method

To validate the accuracy of the vortex panel method model for an infinite wing with sweep, its results are compared to computational fluid dynamics (CFD) simulations over a range of angles of attack and angles of sweep. The infinite wing with sweep is modeled in the CFD simulations by a series of identical, parallel meshes, offset by the sweep angle, with cyclic boundary conditions at the ends.

The CFD simulations were performed in OpenFOAM [2] using the simpleFOAM solver, which implements the SIMPLE finite-volume algorithm for steady-state incompressible flows. In order to model the desired inviscid flows, the solver's turbulence type is set to "laminar", and the simulation is run at a high Reynolds number. The velocity equations use a PBiCG solver with a DILU pre-conditioner, and enforces a slip boundary condition on the airfoil surface. The pressure equations use a PCG solver with a DIC pre-conditioner, with a zero gradient condition on the airfoil surface.

To ensure that the values from the simulations used for the comparison represent fully grid-converged values, three grid densities (herein referred to as "coarse", "medium", and

[2] See the OpenFOAM User Guide at http://www.openfoam.com/documentation/user-guide/ (retrieved Dec. 2018)

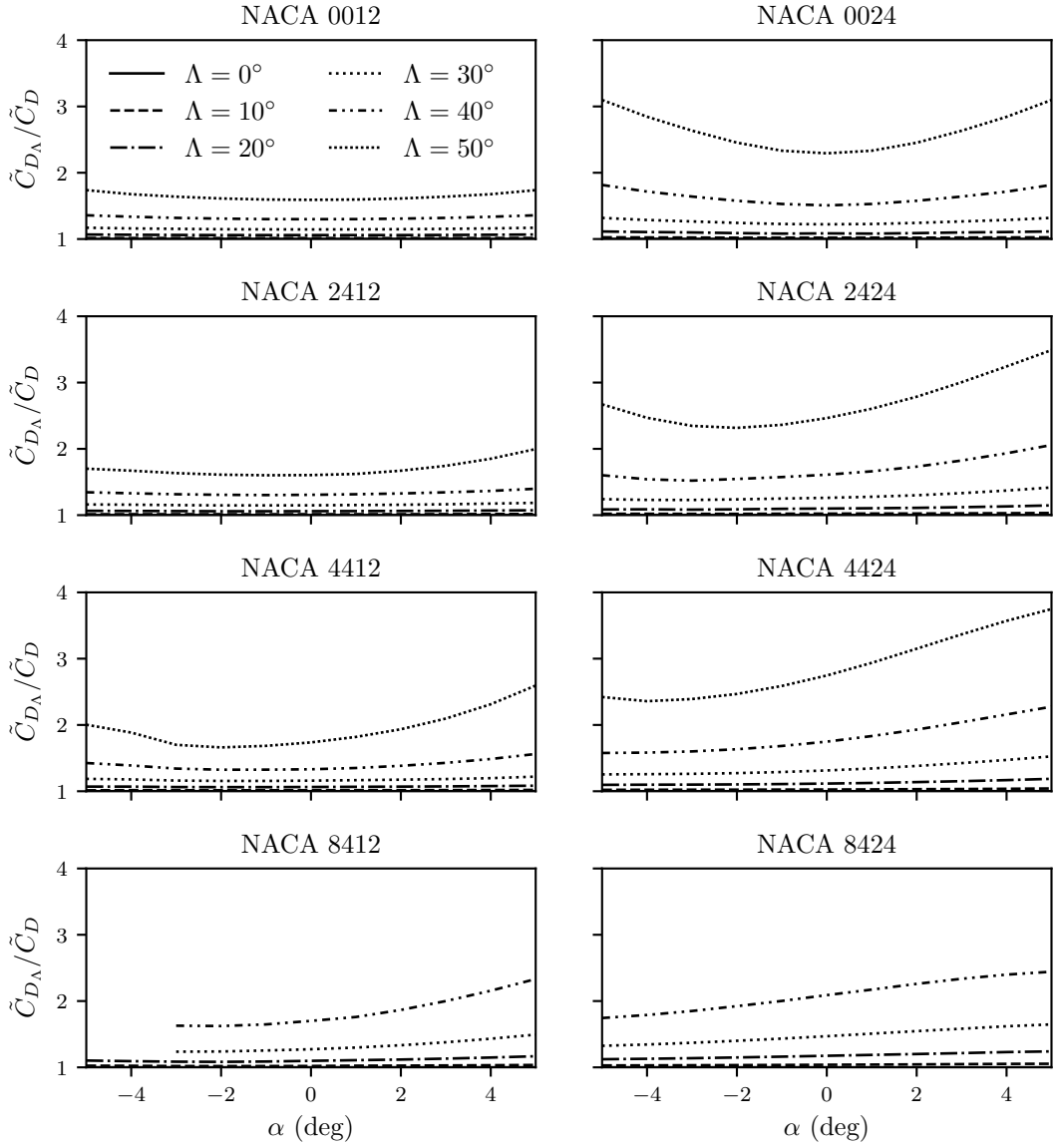

Fig. 5.13: The ratio of the section drag coefficient of an infinite wing with sweep, $\tilde{C}_{D_\Lambda}$, to that of an un-swept infinite wing, $\tilde{C}_D$, as a function of sweep. (Gaps in data exist for non-convergent XFOIL cases)

"fine") are used. The medium grid is created by removing every-other node from the fine grid, and the coarse grid is created by removing every-other node from the medium grid. This process ensures a uniform refinement throughout the computational domain. The coarse grid is shown in Fig. 5.14. The results from these three grids are extrapolated using Richardson extrapolation (see Appendix A) to determine an apparent order of convergence

and predict the values produced by a grid with an infinite number of nodes [21,50]. Table 5.1 shows the flow properties used in the simulations, as well as the node counts for the fine grid.

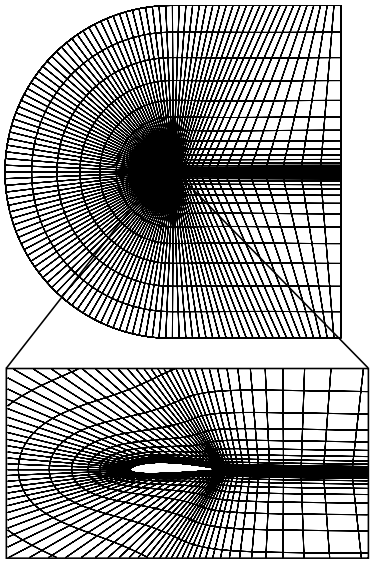

Fig. 5.14: Schematic of the coarse CFD grid. The infinite wing with sweep is created by offsetting copies of this 2D grid along the ζ-axis.

Table 5.1: Flow properties and node counts used in CFD validation simulations.

Simulation Properties	Fine Grid
Airfoil	NACA 2412
Airfoil Surface Cells	208
Wake Cells	152
Radial Cells	152
Spanwise Cells	16
Total Cells	1,245,184

The results of the grid-convergence study for the 20° sweep case are shown in Figs. 5.15–5.17. The grid convergence is best seen in the plot of the drag coefficient in Fig. 5.17, where there is a clear convergence towards a drag coefficient of zero, the anticipated result for inviscid simulations.

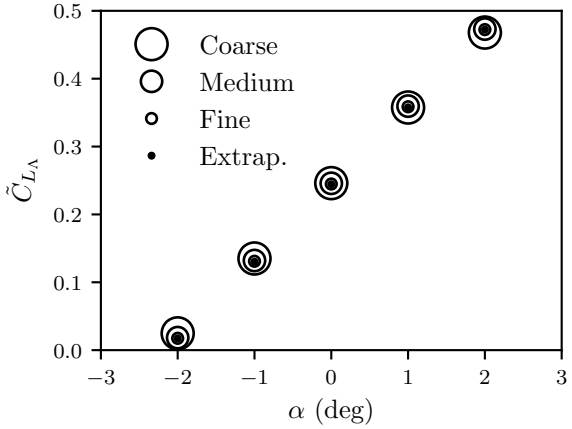

Fig. 5.15: Convergence of the section lift coefficient.

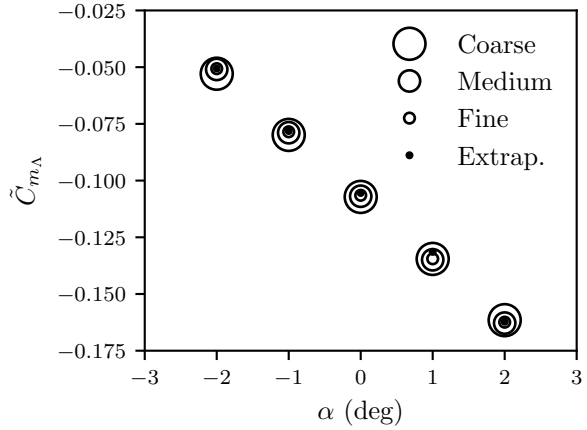

Fig. 5.16: Convergence of the section moment coefficient about the swept leading edge.

5.9.1 Validation of the Lift Model

The CFD results obtained using the fine grid are treated as the "true" values for the infinite wing section properties, and are the baseline against which the results of thin-airfoil theory and the vortex panel method are compared in Figs. 5.18 and 5.19. Figure 5.18 shows the change in section-lift coefficient with angle of attack for an infinite wing with a NACA 2412 airfoil section and six different sweep values, and Fig. 5.19 shows the error of each approximation method for each of those cases.

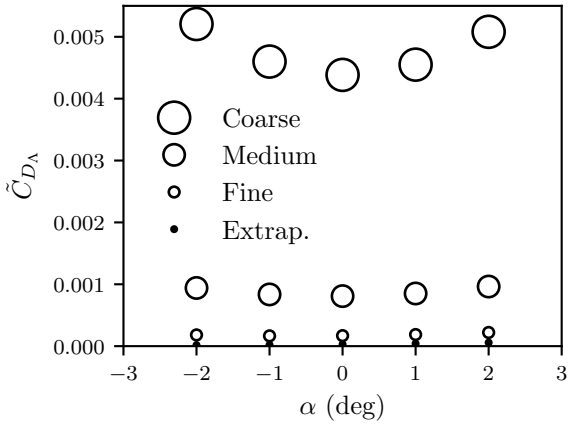

Fig. 5.17: Convergence of the section drag coefficient.

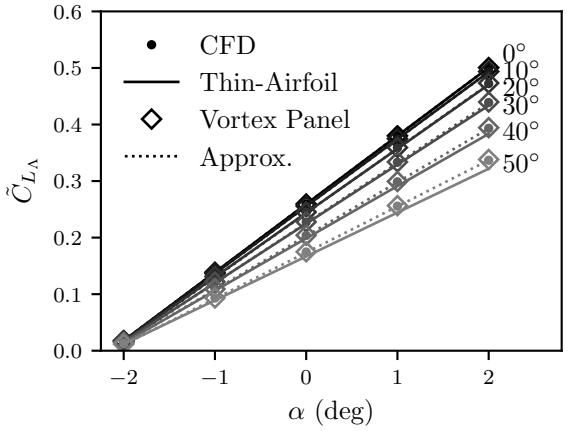

Fig. 5.18: The section coefficient of lift for a NACA 2412 infinite wing with sweep.

The approximation of the section-lift coefficient obtained from Eq. (5.44) is also shown in Fig. 5.18, and its error is compared in Fig. 5.19. The value of $R_{\tilde{C}_{L,\alpha}}$ used in Eq. (5.44) is found using the curve fit described by Eq. (5.61), and $\Delta\alpha_{L0}$ is described by Eq. (5.63). The resulting approximation for the section-lift coefficient is a function solely of freestream properties, sweep angle, and properties of the unswept airfoil

$$\frac{\tilde{C}_{L_\Lambda}}{\tilde{C}_L} = \frac{\sqrt{\cos^2\alpha\cos^2(\Lambda-\beta)+\sin^2\alpha\cos^2\beta}}{\sqrt{1-\sin^2\alpha\sin^2\beta}}\frac{\cos\beta_\Lambda}{\sqrt{1-\sin^2\alpha_\Lambda\sin^2\beta_\Lambda}}R_{\tilde{C}_{L,\alpha}}\frac{(\alpha_\Lambda-\alpha_{L0}-\Delta\alpha_{L0})}{(\alpha-\alpha_{L0})} \tag{5.69}$$

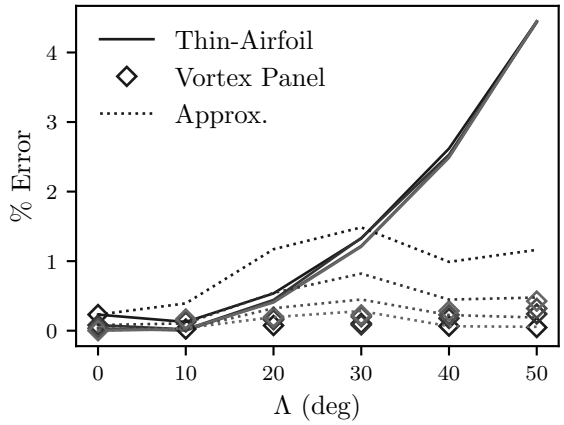

Fig. 5.19: The percent error of the approximation in Eq. (5.44), the thin-airfoil theory approximation, and vortex panel method approximation for the section-lift coefficient. Each curve represents an angle of attack shown in Fig. 5.18.

Note that, in Figs. 5.18 and 5.19, CFD, thin-airfoil theory, the vortex panel method, and Eq. (5.69) all predict a reduction in the lift slope as sweep increases. At higher sweep angles, however, it becomes clear that thin-airfoil theory over-predicts the section lift-slope reduction. This trend is made most apparent in Fig. 5.19. At low angles of sweep the difference between all prediction methods is small. As more sweep is introduced, the vortex panel method and the approximation described by Eq. (5.69) maintain their level of error, but the thin-airfoil theory prediction diverges. Note that the -2° angle of attack case was omitted from Fig. 5.19 due to the fact that the section-lift coefficient is near zero at -2° angle of attack, so small deviations result in large changes in percent error.

5.9.2 Validation of the Moment Model

The moment data obtained from the vortex panel method, along with the approximation of Eq. (5.54), is compared against the CFD simulation results in Figs. 5.20 and 5.21. The results shown in these figures is recognizably similar to those shown in Figs. 5.18 and 5.19. The vortex panel method approximation closely replicates the data of the CFD simulations, and the approximation of Eq. (5.54) shows a level of accuracy just better than the thin-airfoil theory approximation. There is more spread in the error associated with the

moment predictions, but it is on the order of the error in the lift approximations, less than 4%.

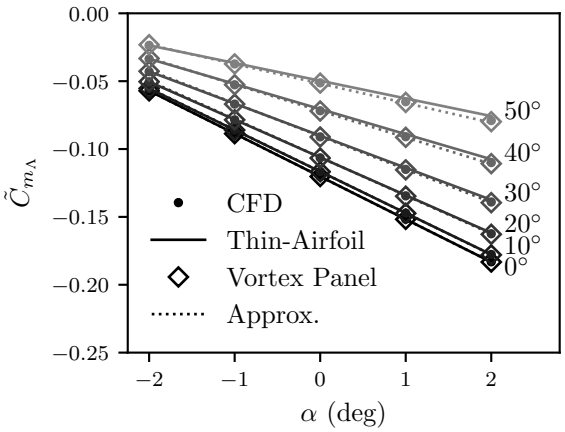

Fig. 5.20: The section moment coefficient for a NACA 2412 infinite wing with sweep.

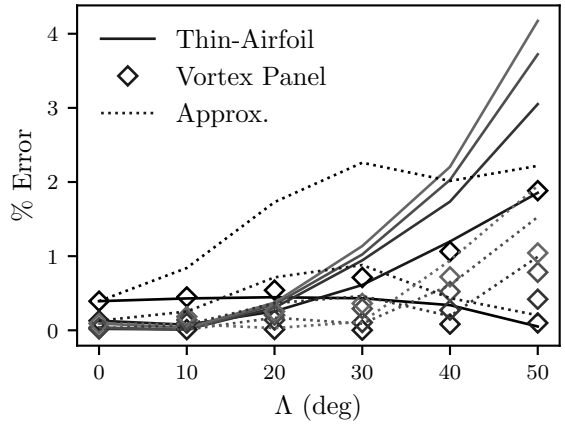

Fig. 5.21: The percent error of the approximation in Eq. (5.54), the thin-airfoil theory approximation, and vortex panel method approximation for the section-moment coefficient. Each of the curves represents an angle of attack shown in Fig. 5.20.

5.10 Conclusion

Thin-airfoil theory predicts a reduction of section lift when sweep is applied to an infinite wing. This approximation for the reduction in section lift of an infinite wing, described by Eq. (5.6), is limited to thin airfoils, small angles of attack, and flows without side-slip. These restrictions are the result of the assumption that the total section circulation produced by an infinite wing with sweep is the same as the section circulation produced by the corresponding un-swept wing. When relaxing this assumption, it is found that, as sweep is introduced to an infinite wing, the effective freestream velocity and angle of attack change, as described in Eqs. (5.29) and (5.30), along with the effective airfoil geometry. By characterizing those changes, a prediction for the change in section lift coefficient is defined in Eq. (5.35).

In the more-general prediction given in Eq. (5.35), the effective airfoil's section circulation, Γ_Λ, is unknown. Accordingly, it is predicted using a vortex panel method, with the application of the effective freestream velocity, angle of attack, and airfoil geometry. Using the vortex panel method, the effective section circulation may either be directly evaluated or approximated by fitting Eq. (5.41) to the resulting predictions. Good agreement is seen when comparing the results of the vortex panel method to the section lift and section moment predictions made by CFD simulations, as observed in Figs. 5.18–5.21. This level of agreement proves the usefulness of these models in predicting the swept section properties necessary for the general implementation of lifting-line theory.

CHAPTER 6

A GENERAL IMPLEMENTATION OF LIFTING-LINE THEORY

Having considered the velocity induced along the locus of aerodynamic centers in Chapter 2, the shape of the locus of aerodynamic centers itself in Chapter 4, and the section properties of swept wings in Chapter 5, a more general implementation of lifting-line theory can be developed. First, following Prandtl's original methodology in Eqs. (1.1) through (1.9), but allowing for generalizations of freestream direction and wing geometry, an analytic implementation is considered. Then, the advantages of a numerical implementation are also examined.

6.1 General, Analytic Lifting-Line Theory Implementation

As discussed in Chapter 2, the velocity induced at a point, z_0, along the locus of aerodynamic centers is characterized by Eq. (2.43) as

$$\vec{V}_i(z_0) = \int_{-b/2}^{b/2} \left(d\vec{V}_{\mathrm{LAC}}(z_0) + d\vec{V}_{\mathrm{TV}_\delta}(z_0) \right) dz$$

which is the sum of the influences of the bound vortex filament, placed along the locus of aerodynamic centers, and the jointed trailing vortex sheet. The local velocity at z_0 is thus the sum of the freestream velocity and the induced velocity

$$\vec{V}(z_0) = \vec{V}_\infty + \vec{V}_i(z_0) = \left[V_{\infty_x} + V_{i_x}(z_0),\, V_{\infty_y} + V_{i_y}(z_0),\, V_{\infty_z} + V_{i_z}(z_0) \right] \tag{6.1}$$

Unfortunately, the integral describing $\vec{V}_i(z_0)$ in Eq. (2.43) is beyond the scope of traditional analytic integration methods for an arbitrary wing, and, because of the singularity contained within the bounds of integration, numerical evaluation of that integral does not yield usable results. Even using techniques to improve its numerical behavior, the integral

does not have good numerical performance [51]. But, before resulting to a numerical approximation for the induced velocities, the simplest extension of Prandtl's implementation is considered, that of a straight wing in sideslip.

Using Eq. (5.30), the effective angle of attack, taking into account the induced velocity, for each spanwise section can be written as

$$\alpha_\Lambda(z_0) = \tan^{-1}\Big(\frac{\tan\alpha_i(z_0)\cos\beta_i(z_0)}{\cos\big(\Lambda(z_0)-\beta_i(z_0)\big)}\Big) \tag{6.2}$$

where

$$\alpha_i(z_0) = \tan^{-1}\Big(\frac{V_{\infty_y}+V_{i_y}(z_0)}{V_{\infty_x}+V_{i_x}(z_0)}\Big) \tag{6.3}$$

$$\beta_i(z_0) = \tan^{-1}\Big(\frac{V_{\infty_z}+V_{i_z}(z_0)}{V_{\infty_x}+V_{i_x}(z_0)}\Big) \tag{6.4}$$

Rearranging, the effective angle of attack in Eq. (6.2) can be expressed in the form

$$\alpha_\Lambda(z_0) = \tan^{-1}\left(\frac{V_{\infty_y}+V_{i_y}(z_0)}{\big(V_{\infty_x}+V_{i_x}(z_0)\big)\cos\Lambda(z_0)+\big(V_{\infty_z}+V_{i_z}(z_0)\big)\sin\Lambda(z_0)}\right) \tag{6.5}$$

With an expression for the effective angle of attack, a formulation similar to Prandtl's can be obtained.

The lift coefficient produced by each swept section can be modeled as the linear function of angle of attack derived from Eq. (5.43)

$$\tilde{C}_{L_\Lambda}(z_0) = R_\infty(z_0)R_{\Lambda_\infty}(z_0)\tilde{C}_{L_\Lambda,\alpha_\Lambda}(z_0)\Big(\alpha_\Lambda(z_0)-\alpha_{L0_\Lambda}(z_0)\Big) \tag{6.6}$$

with

$$R_\infty(z_0) = \frac{\sqrt{\cos^2\alpha_\infty\cos^2(\Lambda(z_0)-\beta_\infty)+\sin^2\alpha_\infty\cos^2\beta_\infty}}{\sqrt{1-\sin^2\alpha_\infty\sin^2\beta_\infty}}$$

$$R_{\Lambda_\infty}(z_0) = \frac{\cos\beta_{\Lambda_\infty}(z_0)}{\sqrt{1-\sin^2\alpha_{\Lambda_\infty}(z_0)\sin^2\beta_{\Lambda_\infty}(z_0)}}$$

where $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$ is the effective lift slope of the effective airfoil and α_{L0_Λ} is the effective zero-lift angle of attack of the airfoil. The angles α_∞, β_∞, α_{Λ_∞}, and β_{Λ_∞} are the aerodynamic angles

and the effective aerodynamic angles, respectively, of the section, calculated neglecting the induced velocity. The induced velocities are neglected in these angles to linearize Eq. (6.6).

The section lift coefficient is also related to the local circulation by the form of the Kutta-Joukowski theorem given in Eq. (5.34)

$$\tilde{C}_{L_\Lambda}(z_0) = R_\infty(z_0)\frac{2\Gamma(z_0)}{V_\infty c(z_0)\cos\Lambda(z_0)} \tag{6.7}$$

Equating Eq. (6.6) and Eq. (6.7), and using the definition for α_Λ from Eq. (6.5), yields

$$\tan^{-1}\left(\frac{V_{\infty_y} + V_{i_y}(z_0)}{(V_{\infty_x} + V_{i_x}(z_0))\cos\Lambda(z_0) + (V_{\infty_z} + V_{i_z}(z_0))\sin\Lambda(z_0)}\right) - \alpha_{L0_\Lambda}(z_0) = \frac{2\Gamma(z_0)}{R_{\Lambda_\infty}(z_0)\tilde{C}_{L_\Lambda,\alpha_\Lambda}(z_0)V_\infty c(z_0)\cos\Lambda(z_0)} \tag{6.8}$$

where V_{i_x}, V_{i_y}, and V_{i_z} are functions of the circulation distribution, $\Gamma(z)$. As in Prandtl's implementation, the change of variables defined by Eqs. (1.5) and (1.6) can be used to express $\Gamma(z)$ as the Fourier sine series shown in Eqs. (1.7) and (1.8). To determine the Fourier coefficients, A_n, Eq. (6.8) is evaluated at N control points along the wing, resulting in a system of N non-linear equations to be solved, after performing the integration necessary to determine the induced velocity, $\vec{V}_i$.

Assuming that α_Λ is small and that V_{i_y} is the only non-negligible component of the induced velocity, as did Prandtl, the effective angle of attack described in Eq. (6.5) can be written as

$$\alpha_\Lambda(z_0) \approx \frac{V_{\infty_y} + V_{i_y}(z_0)}{V_{\infty_x}\cos\Lambda(z_0) + V_{\infty_z}\sin\Lambda(z_0)} \tag{6.9}$$

and Eq. (6.8) becomes

$$\frac{V_{\infty_y} + V_{i_y}(z_0)}{V_{\infty_x}\cos\Lambda(z_0) + V_{\infty_z}\sin\Lambda(z_0)} - \alpha_{L0_\Lambda}(z_0) = \frac{2\Gamma(z_0)}{R_{\Lambda_\infty}(z_0)\tilde{C}_{L_\Lambda,\alpha_\Lambda}(z_0)V_\infty c(z_0)\cos\Lambda(z_0)} \tag{6.10}$$

where

$$V_{i_y}(z_0) = \int_{-b/2}^{b/2}\left(dV_{\mathrm{LAC}_y}(z_0) + dV_{\mathrm{TV}_{\delta y}}(z_0)\right)dz \tag{6.11}$$

Equation (6.10) can be rewritten in terms of θ using Eqs. (1.5)–(1.8), yielding

$$\begin{aligned}\frac{V_{\infty_y}}{V_{\infty_x}\cos\Lambda(\theta_0)+V_{\infty_z}\sin\Lambda(\theta_0)}&-\alpha_{L0_\Lambda}(\theta_0)\\ &=2bV_\infty\sum_1^N A_n\left(\frac{2\sin(n\theta_0)}{R_{\Lambda_\infty}(\theta_0)\tilde{C}_{L_\Lambda,\alpha\Lambda}(\theta_0)V_\infty c(\theta_0)\cos\Lambda(\theta_0)}\right.\\ &\qquad\left.-\frac{\int_0^\pi\left(\frac{b}{2}\sin(\theta)\sin(n\theta)\frac{dV_{\mathrm{LAC}_y}(\theta_0)}{\Gamma(\theta)}+n\cos(n\theta)\frac{dV_{\mathrm{TV}_{\delta y}}(\theta_0)}{\Gamma'(\theta)}\right)d\theta}{V_{\infty_x}\cos\Lambda(\theta_0)+V_{\infty_z}\sin\Lambda(\theta_0)}\right)\end{aligned}\tag{6.12}$$

Now, when Eq. (6.12) is evaluated at N control points along the wing, the result is a linear system of N equations whose solution is the Fourier coefficients, A_n.

In the case of a straight wing in side-slip and zero angle of attack, the locus of aerodynamic centers is approximated as a line along the z-axis ($f(z)=0, f'(z)=0$). Thus, the bound vortex filament induces no velocity along the locus of aerodynamic centers, $dV_{\mathrm{LAC}_y}=0$, and Eq. (6.12) becomes

$$\begin{aligned}\frac{V_{\infty_y}}{V_{\infty_x}\cos\Lambda(\theta_0)+V_{\infty_z}\sin\Lambda(\theta_0)}&-\alpha_{L0_\Lambda}(\theta_0)\\ &=2bV_\infty\sum_1^N A_n\left(\frac{2\sin(n\theta_0)}{R_{\Lambda_\infty}(\theta_0)\tilde{C}_{L_\Lambda,\alpha\Lambda}(\theta_0)V_\infty c(\theta_0)\cos\Lambda(\theta_0)}\right.\\ &\qquad\left.-\frac{\frac{n}{4\pi}\int_0^\pi\frac{\cos(n\theta)}{|\vec{r}-\vec{r}_\delta|}\left(\frac{\delta}{r}-\frac{u_z\delta+u_x\frac{b}{2}\left(\cos(\theta)-\cos(\theta_0)\right)}{\left(|\vec{r}-\vec{r}_\delta|-u_z\frac{b}{2}\left(\cos(\theta)-\cos(\theta_0)\right)+u_x\delta\right)}\right)d\theta}{V_{\infty_x}\cos\Lambda(\theta_0)+V_{\infty_z}\sin\Lambda(\theta_0)}\right)\end{aligned}\tag{6.13}$$

where

$$r=\frac{b}{2}\left|\cos(\theta)-\cos(\theta_0)\right|\tag{6.14}$$

$$|\vec{r}-\vec{r}_\delta|=\sqrt{\frac{b^2}{4}\left(\cos(\theta)-\cos(\theta_0)\right)^2+\delta^2}\tag{6.15}$$

Even with the small-angle approximation and the simplified case described by Eq. (6.13), the analytic application of general lifting-line theory does not result in a useful formulation, due to the complexity of the integration required to calculate the induced velocities. Therefore, it is convenient to apply the general considerations for the velocity induced along the locus of aerodynamic centers, the shape of the locus of aerodynamic centers itself, and the

section properties of swept wings to the discretized methodology presented by Phillips [19], and shown in Eqs. (1.12)–(1.15), since it does not require explicit integration.

6.2 General, Numerical Lifting-Line Theory Implementation

The numerical implementation of lifting-line theory developed by Phillips will be the foundation for the general lifting-line theory implementation [5, 19]. Phillips' implementation separates the bound vortex filament and trailing vortex sheet into a discrete number of abutted *horseshoe vortices*, each consisting of a constant-strength vortex segment and two semi-infinite vortices, shown in Fig. 6.1. Endpoints of the bound portion of each horseshoe vortex lay on the wing's locus of aerodynamic centers, and the trailing portion of each horseshoe vortex is aligned with the freestream.

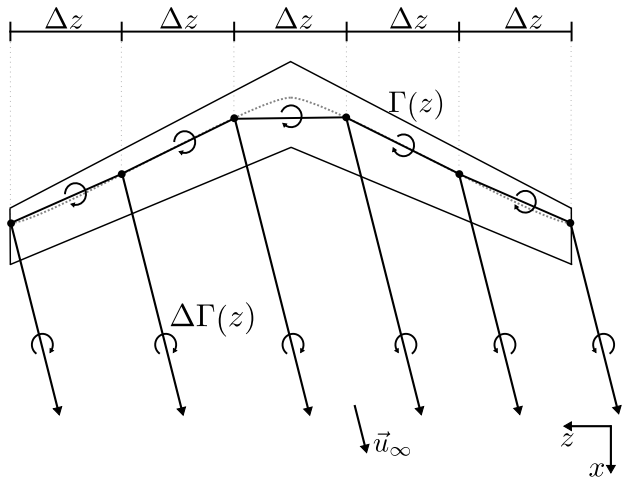

Fig. 6.1: A wing whose circulation is approximated by a finite number of horseshoe vortices.

It is heretofore shown in Sections 2.1.1 and 2.2.1 that Phillips' numerical implementation of lifting-line theory is limited in similar ways as Prandtl's classic implementation. Its advantage to this work, however, lies in the fact that no integration is required in its application, in contrast to the implementation derived in Section 6.1. Thus, by applying the concepts of conditional concavity and jointed trailing vortices, a general, numerical lifting-line theory implementation is obtained.

The local velocity is calculated at a control point located on each bound vortex segment with the equation

$$\vec{V}_i = \vec{V}_\infty + \sum_{j=1}^{N} \Gamma_j \vec{v}_{ji} \tag{6.16}$$

where Γ_j is the strength of each horseshoe vortex, and $\vec{v}_{ji}$ is the influence of vortex j on control point i. Using the notation in Fig. 6.2, and Eqs. (2.5) and (2.19), $\vec{v}_{ji}$ can be described by the equation

$$\vec{v}_{12} = \frac{1}{4\pi}\Bigg(- \frac{\vec{u}_\infty \times \vec{r}_{a'}}{r_{a'}\big(r_{a'} - \vec{u}_\infty \cdot \vec{r}_{a'}\big)} + \frac{(r_{a'} + r_a)(\vec{r}_{a'} \times \vec{r}_a)}{r_{a'} r_a (r_{a'} r_a + \vec{r}_{a'} \cdot \vec{r}_a)} + \frac{(r_a + r_b)(\vec{r}_a \times \vec{r}_b)}{r_a r_b (r_a r_b + \vec{r}_a \cdot \vec{r}_b)} + \frac{(r_b + r_{b'})(\vec{r}_b \times \vec{r}_{b'})}{r_b r_{b'} (r_b r_{b'} + \vec{r}_b \cdot \vec{r}_{b'})} + \frac{\vec{u}_\infty \times \vec{r}_{b'}}{r_{b'}\big(r_{b'} - \vec{u}_\infty \cdot \vec{r}_{b'}\big)} \Bigg) \tag{6.17}$$

where

$$\vec{r}_a = \Big[\tilde{f}_{z_2}(z_2) - \tilde{f}_{z_2}(z_a), 0\,, z_2 - z_a\Big]$$

$$\vec{r}_{a'} = \Big[\tilde{f}_{z_2}(z_2) - \tilde{f}_{z_2}(z_a) - \frac{\delta}{\sqrt{1 + \tilde{f}'_{z_2}(z_a)^2}}, 0\,, z_2 - z_a + \frac{\delta \tilde{f}'_{z_2}(z_a)}{\sqrt{1 + \tilde{f}'_{z_2}(z_a)^2}}\Big]$$

$$\vec{r}_b = \Big[\tilde{f}_{z_2}(z_2) - \tilde{f}_{z_2}(z_b), 0\,, z_2 - z_b\Big]$$

$$\vec{r}_{b'} = \Big[\tilde{f}_{z_2}(z_2) - \tilde{f}_{z_2}(z_b) - \frac{\delta}{\sqrt{1 + \tilde{f}'_{z_2}(z_b)^2}}, 0\,, z_2 - z_b + \frac{\delta \tilde{f}'_{z_2}(z_b)}{\sqrt{1 + \tilde{f}'_{z_2}(z_b)^2}}\Big]$$

Here, $\tilde{f}_{z_2}$ and $\tilde{f}'_{z_2}$ are defined by Eqs. (2.13) and (2.14) and the locus of aerodynamic centers is defined using Eq. (4.5).

Note that, as suggested by Phillips [5], in the case where $i = j$, the third term on the right-hand-side of Eq. (6.17) (i.e. the term describing the influence of the bound vortex segment) should be omitted from the calculation, because the control point is located on the bound vortex segment itself. This term should analytically result in zero induced velocity, because the straight vortex segment does not induce any velocity along its length, but, in practice, round-off errors in the control point's location result in high, non-zero velocities.

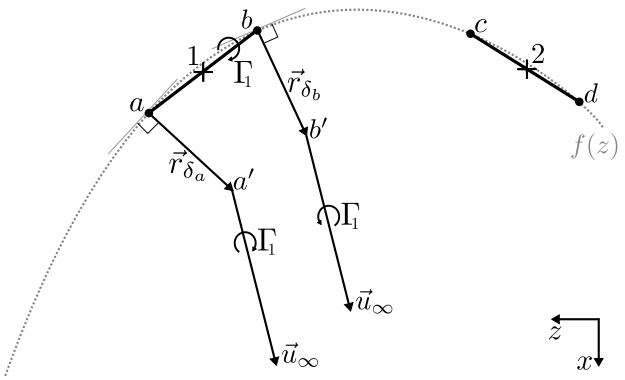

Fig. 6.2: The geometry used to define the velocity induced at a point by the jointed horseshoe vortex 1, on the control point 2.

Equating the vectorized form of the Kutta-Joukowski law [5, 7, 8, 19]

$$d\vec{F}_i = \rho\Gamma_i\vec{V}_i \times d\vec{l}_i \tag{6.18}$$

to section lift information—obtained from an analytic, numerical, empirical, or experimental prediction—as a function of the local velocity, $\tilde{C}_{L_\Lambda}(\vec{V}_i)$, results in a non-linear system of equations which can be solved iteratively

$$\rho\Gamma_i\left|\left(\vec{V}_\infty + \sum_{j=1}^{N}\Gamma_j\,\vec{v}_{ji}\right)\times d\vec{l}_i\right| - \frac{1}{2}\rho V_\infty^2\tilde{C}_{L_{\Lambda i}}(\vec{V}_i)dA_i = 0 \tag{6.19}$$

Recall from Chapter 5 that the aerodynamic properties and effective flow properties of a swept wing section vary as a function of the sweep angle, Λ, and several predictions for $\tilde{C}_{L_\Lambda}(\vec{V}_i)$ are described in Eqs. (5.6), (5.35), and (5.44). Slices of the wing are taken normal to the locus of aerodynamic centers, $f(z)$, such that each section of the wing has a local, effective sweep angle, related to the locus of aerodynamic centers according to

$$\Lambda(z) = -\tan^{-1}\Big(f'(z)\Big) \tag{6.20}$$

Note that this definition of effective sweep angle results in a negative effective sweep when

$f'(z) > 0$. This sign is important in accounting for the effective sideslip experienced by a section.

It is convenient to rewrite Eq. (6.19) in the non-dimensionalized form

$$2G_i \left| \left(\vec{u}_\infty + \sum_{j=1}^{N} G_j \, \vec{v}_{ji} \right) \times \vec{\zeta}_i \right| - \tilde{C}_{L_{\Lambda i}}(\vec{V}_i) = 0 \tag{6.21}$$

where

$$\vec{u}_\infty = \frac{\vec{V}_\infty}{V_\infty}, \quad G_i = \frac{\Gamma_i}{V_\infty}, \quad \vec{\zeta}_i = \frac{d\vec{l}_i}{dA_i} \tag{6.22}$$

The non-linear system described by Eq. (6.21) can be solved using Newton's method by writing the system in the form [5]

$$\vec{\mathrm{f}}(\vec{G}) = \vec{R} \tag{6.23}$$

where

$$\mathrm{f}_i(\vec{G}) = 2G_i \left| \left(\vec{u}_\infty + \sum_{j=1}^{N} G_j \, \vec{v}_{ji} \right) \times \vec{\zeta}_i \right| - \tilde{C}_{L_{\Lambda i}} \tag{6.24}$$

and $\vec{R}$ is a vector of residuals. The solution to Eq. (6.23) is the vector of normalized vortex strengths, $\vec{G}$, that results in a residual vector equal to zero.

Newton's method begins with an initial guess of the solution vector. One initial guess that has resulted in good convergence for the cases performed herein, is given by the function

$$G_i = \frac{1}{2} c_r \cos\Lambda_w C_{L,\alpha_r} \left(\frac{V_{\infty_y}}{V_{\infty_x}} - \alpha_{L0_r} \right) \left(1 - \left(\frac{2z_i}{b} \right)^4 \right)^{1/4} \tag{6.25}$$

where c_r, C_{L,α_r}, and α_{L0_r} are the chord, lift slope, and zero-lift angle of attack of the root airfoil, and Λ_w is the sweep of the wing. To identify the solution vector, Newton's method computes an improvement upon the initial guess of $\vec{G}$ such that

$$\bar{J}\Delta\vec{G} = -\vec{R} \tag{6.26}$$

where $\bar{J}$ is an $N \times N$ Jacobian matrix of partial derivatives. The Jacobian matrix is found by differentiating Eq. (6.21), giving

$$\bar{J}_{ij} = \frac{\partial f_i}{\partial G_j} = \begin{cases} 2G_i \frac{\vec{u}_i \times \vec{\zeta}_i}{\left|\vec{u}_i \times \vec{\zeta}_i\right|} \cdot \left(\vec{v}_{ji} \times \vec{\zeta}_i\right) - \frac{\partial \tilde{C}_{L_{\Lambda i}}}{\partial G_j} & i \neq j \\ 2\left|\vec{u}_i \times \vec{\zeta}_i\right| + 2G_i \frac{\vec{u}_i \times \vec{\zeta}_i}{\left|\vec{u}_i \times \vec{\zeta}_i\right|} \cdot \left(\vec{v}_{ji} \times \vec{\zeta}_i\right) - \frac{\partial \tilde{C}_{L_{\Lambda i}}}{\partial G_j} & i = j \end{cases} \tag{6.27}$$

where

$$\vec{u}_i = [u_{i_x}, u_{i_y}, u_{i_z}] = \vec{u}_\infty + \sum_{j=1}^{N} G_j \, \vec{v}_{ji} \tag{6.28}$$

and $\frac{\partial \tilde{C}_{L_{\Lambda i}}}{\partial G_j}$ is found by differentiating Eq. (5.53) with respect to $\vec{G}$

$$\begin{aligned} \frac{\partial \tilde{C}_{L_{\Lambda i}}}{\partial G_j} =& \frac{\partial \left(R_i R_{\Lambda_i} \tilde{C}_{L_\Lambda, \alpha_{\Lambda i}}(\alpha_{\Lambda i} - \alpha_{L0_{\Lambda i}})\right)}{\partial G_j} \\ =& \frac{\partial R_i}{\partial G_j} R_{\Lambda_i} \tilde{C}_{L_\Lambda, \alpha_{\Lambda i}}(\alpha_{\Lambda i} - \alpha_{L0_{\Lambda i}}) \\ &+ R_i \frac{\partial R_{\Lambda_i}}{\partial G_j} \tilde{C}_{L_\Lambda, \alpha_{\Lambda i}}(\alpha_{\Lambda i} - \alpha_{L0_{\Lambda i}}) \\ &+ R_i R_{\Lambda_i} \tilde{C}_{L_\Lambda, \alpha_{\Lambda i}} \frac{\partial \alpha_{\Lambda i}}{\partial G_j} \end{aligned} \tag{6.29}$$

where

$$R_i = \frac{\sqrt{\cos^2 \alpha_i \cos^2(\Lambda_i - \beta_i) + \sin^2 \alpha_i \cos^2 \beta_i}}{\sqrt{1 - \sin^2 \alpha_i \sin^2 \beta_i}} \tag{6.30}$$

$$R_{\Lambda_i} = \frac{\cos \beta_{\Lambda_i}}{\sqrt{1 - \sin^2 \alpha_{\Lambda_i} \sin^2 \beta_{\Lambda_i}}} \tag{6.31}$$

$$\alpha_i = \tan^{-1}\left(\frac{u_{i_y}}{u_{i_x}}\right) \tag{6.32}$$

$$\beta_i = \tan^{-1}\left(\frac{u_{i_z}}{u_{i_x}}\right) \tag{6.33}$$

$$\alpha_{\Lambda_i} = \tan^{-1}\left(\frac{u_{i_y}}{u_{i_x} \cos \Lambda_i + u_{i_z} \sin \Lambda_i}\right) \tag{6.34}$$

$$\beta_{\Lambda_i} = \tan^{-1}\left(\frac{u_{i_z}}{u_{i_x}}\right) - \Lambda_i \tag{6.35}$$

and

$$\begin{aligned}\frac{\partial R_i}{\partial G_j} =& \frac{\partial R_i}{\partial \alpha_i}\frac{\partial \alpha_i}{\partial G_j} + \frac{\partial R_i}{\partial \beta_i}\frac{\partial \beta_i}{\partial G_j} \\ =& \frac{\sin\alpha_i\cos\alpha_i}{\sqrt{1-\sin^2\alpha_i\sin^2\beta_i}}\left(\frac{\sin^2\beta_i\sqrt{\cos^2\alpha_i\cos^2(\Lambda_i-\beta_i)+\sin^2\alpha_i\cos^2\beta_i}}{1-\sin^2\alpha_i\sin^2\beta_i}\right. \\ &\left. + \frac{\cos^2\beta_i-\cos^2(\Lambda_i-\beta_i)}{\sqrt{\cos^2\alpha_i\cos^2(\Lambda_i-\beta_i)+\sin^2\alpha_i\cos^2\beta_i}}\right)\frac{\partial\alpha_i}{\partial G_j} \\ &+ \frac{1}{\sqrt{1-\sin^2\alpha_i\sin^2\beta_i}}\left(\frac{\sin^2\alpha_i\sin\beta_i\cos\beta_i\sqrt{\cos^2\alpha_i\cos^2(\Lambda_i-\beta_i)+\sin^2\alpha_i\cos^2\beta_i}}{1-\sin^2\alpha_i\sin^2\beta_i}\right. \\ &\left. + \frac{\cos^2\alpha_i\sin(\Lambda_i-\beta_i)\cos(\Lambda_i-\beta_i)-\sin^2\alpha_i\sin\beta_i\cos\beta_i}{\sqrt{\cos^2\alpha_i\cos^2(\Lambda_i-\beta_i)+\sin^2\alpha_i\cos^2\beta_i}}\right)\frac{\partial\beta_i}{\partial G_j}\end{aligned} \tag{6.36}$$

$$\begin{aligned}\frac{\partial R_{\Lambda_i}}{\partial G_j} &= \frac{\partial R_{\Lambda_i}}{\partial \alpha_{\Lambda_i}}\frac{\partial \alpha_{\Lambda_i}}{\partial G_j} + \frac{\partial R_{\Lambda_i}}{\partial \beta_{\Lambda_i}}\frac{\partial \beta_{\Lambda_i}}{\partial G_j} \\ &= \frac{\sin\alpha_{\Lambda_i}\cos\alpha_{\Lambda_i}\sin^2\beta_{\Lambda_i}\cos\beta_{\Lambda_i}}{(1-\sin^2\alpha_{\Lambda_i}\sin^2\beta_{\Lambda_i})^{3/2}}\frac{\partial\alpha_{\Lambda_i}}{\partial G_j} - \frac{\cos^2\alpha_{\Lambda_i}\sin\beta_{\Lambda_i}}{(1-\sin^2\alpha_{\Lambda_i}\sin^2\beta_{\Lambda_i})^{3/2}}\frac{\partial\beta_{\Lambda_i}}{\partial G_j}\end{aligned} \tag{6.37}$$

$$\frac{\partial\alpha_i}{\partial G_j} = \frac{u_{i_x}v_{ji_y}-u_{i_y}v_{ji_x}}{u_{i_y}^2+u_{i_x}^2} \tag{6.38}$$

$$\frac{\partial\beta_i}{\partial G_j} = \frac{\partial\beta_{\Lambda_i}}{\partial G_j} = \frac{u_{i_x}v_{ji_z}-u_{i_z}v_{ji_x}}{u_{i_z}^2+u_{i_x}^2} \tag{6.39}$$

$$\frac{\partial\alpha_{\Lambda_i}}{\partial G_j} = \frac{\Big(u_{i_x}\cos\Lambda_i+u_{i_z}\sin\Lambda_i\Big)v_{ji_y}-u_{i_y}\Big(v_{ji_x}\cos\Lambda_i+v_{ji_z}\sin\Lambda_i\Big)}{u_{i_y}^2+\Big(u_{i_x}\cos\Lambda_i+u_{i_z}\sin\Lambda_i\Big)^2} \tag{6.40}$$

After using the Jacobian to solve Eq. (6.26), $\Delta\vec{G}$ is added the initial guess of $\vec{G}$, with a relaxation factor, to obtain an improved estimate for the vortex strengths. The process is repeated until the residual vector falls below the convergence threshold.

Having solved the non-linear system for G_i, the force distribution can be found using Eq. (6.16) in Eq. (6.18). Similarly, the total force on the wing is found by summing the section forces

$$\vec{F} = \rho\sum_{i=1}^{N}\Gamma_i\Big(\vec{V}_\infty + \sum_{j=1}^{N}\Gamma_j\vec{v}_{ji}\Big)\times d\vec{l_i} \tag{6.41}$$

Non-dimensionalizing the total force gives

$$\frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} = 2\sum_{i=1}^{N} G_i\Big(\vec{u}_\infty + \sum_{j=1}^{N} G_j \vec{v}_{ji}\Big) \times \vec{\zeta}_i \, \frac{dA_i}{A} \tag{6.42}$$

The total force given in Eq. (6.42) can be decomposed into the lift, drag, and sideforce coefficients

$$C_L = \frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} \cdot \frac{\hat{\imath}_z \times \vec{u}_\infty}{\sqrt{u_y^2 + u_x^2}} \tag{6.43}$$

$$C_D = \frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} \cdot \vec{u}_\infty \tag{6.44}$$

$$C_S = \frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} \cdot \frac{\vec{u}_\infty \times (\hat{\imath}_z \times \vec{u}_\infty)}{\sqrt{(u_x u_z)^2 + (u_y u_z)^2 + (u_x^2 + u_y^2)^2}} \tag{6.45}$$

where $\hat{\imath}_z$ is the unit vector defining the z-axis.

A viscous correction can be added to the total force given in Eq. (6.42). If an estimate for the section drag coefficient is known

$$\frac{\vec{F}_v}{\frac{1}{2}\rho V_\infty^2 A} = \frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} + \sum_{i=1}^{N} \tilde{C}_{D_{\Lambda i}}(\vec{V}_i) \frac{dA_i}{A} \vec{u}_{n_i} \tag{6.46}$$

where $\vec{u}_{n_i}$ is the unit vector of the local effective freestream, described in Eq. (5.25).

Similarly, the aerodynamic moment generated by the wing can be described by the equation

$$\vec{M} = \sum_{i=1}^{N} \vec{r}_i \times \vec{F} + d\vec{M}_i \tag{6.47}$$

where $\vec{r}_i$ is a vector from the point about which the total moment, $\vec{M}$, is calculated to the point on section i about which the section moment, $d\vec{M}_i$, is calculated (e.g. leading edge, quarter chord, aerodynamic center, etc.). The section moment can be written as

$$d\vec{M}_i = \frac{1}{2}\rho V_\infty^2 dA_i \, c \, \tilde{C}_{m_{\Lambda i}} d\hat{l} \tag{6.48}$$

where $\hat{dl}$ is the unit vector in the direction of $\vec{dl}$. Equation (6.47) can be non-dimensionalized to give

$$\frac{\vec{M}}{\frac{1}{2}\rho V_\infty^2 A\, l_{\text{ref}}} = \sum_{i=1}^{N} \frac{\vec{r}_i}{l_{\text{ref}}} \times \frac{\vec{F}}{\frac{1}{2}\rho V_\infty^2 A} + \tilde{C}_{m_{\Lambda i}} \frac{dA_i\, c_i}{A\, l_{\text{ref}}} \hat{dl} \tag{6.49}$$

In Eqs. (6.47) and (6.49), $\vec{F}$ may be replaced with $\vec{F}_v$ to account for viscous effects.

This implementation of lifting-line theory can be applied to the more-general cases of wings with sweep, and wings in sideslip. This general applicability is validated in the following chapter, along with a study of the effects of $\Delta\bar{z}$ and δ/c on the solution. Source code applying the general implementation of lifting-line theory is given in Appendix B.2.

CHAPTER 7

VALIDATION OF THE GENERAL IMPLEMENTATION

Consider six methods for predicting the lift produced by a wing: the general lifting-line theory implementation derived in Chapter 6; Phillips' implementation of lifting-line theory [19]; a modified version of Phillips' implementation that utilizes the finite-core vortex model defined in Eq. (1.11) [12]; an application of Weissinger's lifting-line theory implementation [13, 17]; PAN AIR, a widely-used high-order panel method [52]; and experimental results from Weber and Brebner [20]. In this chapter, these six methods will be compared and analyzed to validate the convergence, accuracy, and sensitivity of the general implementation of lifting-line theory.

The wing used in the analysis shown in Fig. (1.5), for which experimental results are readily available from Weber and Brebner [20], will be used for many of the analyses performed in this chapter. Recall that it is a rectangular wing with 45° sweep and a symmetric airfoil with 12% thickness. In the predictions of this wing's aerodynamic properties by the general lifting-line theory implementation, Phillips' implementation, and the finite-core implementation, 160 nodes are used (cosine-clustered as suggested by Phillips [19]), along with the effective airfoil properties of a NACA 0012 and Küchemann's model for the locus of aerodynamic centers, given in Eq. (4.5). The finite core radius, r_c/c, used in the finite-core implementation of lifting-line theory was calculated as suggested by Ashenberg and Weihs [9], resulting in $r_c/c = 0.15$ for this wing. The same number of spanwise, cosine-clustered nodes are used in Weissinger's method, but the section properties are dictated by the camber-line slope at the 3/4 chord location (i.e. zero for the NACA 0012 airfoil), and the locus of aerodynamic centers is assumed to follow the 1/4 chord line. The results obtained from PAN AIR are calculated using a mesh resolved in both the chordwise and spanwise directions. To maintain consistency with the other inviscid methods, the endcaps of the PAN AIR simulations were truncated in the results.

7.1 Convergence

The convergence of the general lifting-line implementation is observed by comparing the predictions made by the implementation as the node count increases, using the technique of Richardson extrapolation detailed in Appendix A. The convergence is first analyzed for the case of a straight wing in sideslip, while varying the joint length, to isolate the effect of δ/c on convergence. Using the value for δ/c resulting from that study, the convergence for a range of swept wings in a variety of flow conditions is considered, while varying the blending length, $\Delta\bar{z}$.

7.1.1 Effect of Joint Length

To isolate the effect of the trailing vortex joint length, δ/c, on the convergence of the general lifting-line theory implementation, the case of a straight wing—with a NACA 2412 airfoil—in sideslip is considered. Because the locus of aerodynamic centers is a straight line for a straight, rectangular wing (see Eq.(4.5)), $\Delta\bar{z}$ does not affect the solution in this case. Therefore, the lifting-line implementation's sensitivity to δ/c can be explored, unaffected by the blending length.

Table 7.1 shows the parameters that are varied to study the effect of the joint length. Each parameter is sampled at 10 uniformly spaced values along its designated range, while all others are held at a "standard" value. For each case, the convergence rates of the lift coefficient and the RMS change in circulation distribution are recorded. The results are shown in Figs. 7.1–7.4. For each wing, the convergence is calculated using both a uniform distribution of nodes (dashed lines) and a cosine-clustered distribution (solid lines) [19].

Table 7.1: Summary of the parameters varied to observe the effect of joint length on convergence.

Parameter	Min. Value	Max. Value	Std. Value
δ/c	0	1	—
β	0°	45°	30°
α	−5°	10°	5°
R_A	2	24	8
R_T	0.1	1	1

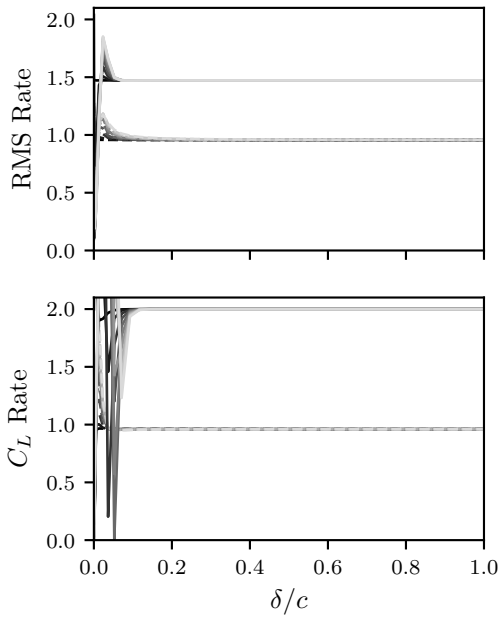

Fig. 7.1: The sensitivity of the general lifting-line implementation's convergence rate to joint length and side-slip angle, dark ($\beta_{\min}$) to light ($\beta_{\max}$).

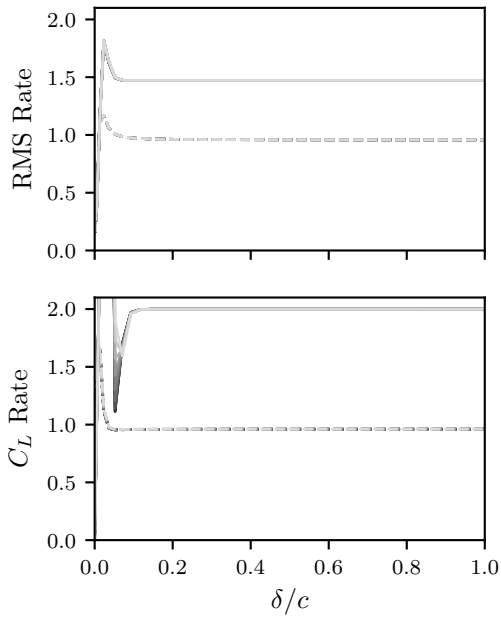

Fig. 7.2: The sensitivity of the general lifting-line implementation's convergence rate to joint length and angle of attack, dark ($\alpha_{\min}$) to light ($\alpha_{\max}$).

From the results shown in Figs. 7.1–7.4, it can be observed that convergence of the general lifting-line theory implementation remains largely unaffected once δ/c is above a value of approximately 0.15. The convergence rates for both the RMS and C_L demonstrate first-order convergence when using a uniform node distribution, but when using a cosine-clustered distribution the RMS convergence rate increases to 1.5 and the convergence rate of C_L increases to two. Additionally, above this joint length threshold, the convergence of the implementation is independent of angle of attack and side-slip and only demonstrates slight variations with changes in aspect ratio and taper ratio.

The first-order convergence observed in the studies with uniform node spacing is consistent with the first-order approximations made in the numerical implementation of lifting-line theory (i.e. the continuous trailing vortex sheet and variable-strength bound vortex are replaced by a set of constant-strength horseshoe vortices), and the first order integration used in calculating the total lift (i.e. Eq. (1.13)). The clustering of the nodes in areas

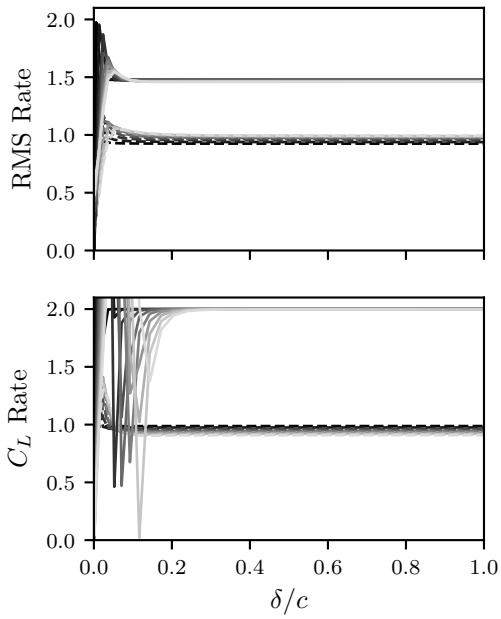

Fig. 7.3: The sensitivity of the general lifting-line implementation's convergence rate to joint length and aspect ratio, dark ($R_{A_{\min}}$) to light ($R_{A_{\max}}$).

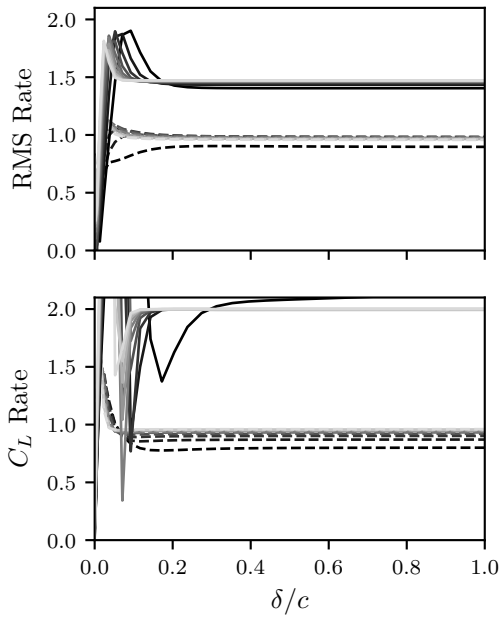

Fig. 7.4: The sensitivity of the general lifting-line implementation's convergence and results to joint length and taper ratio, dark ($R_{T_{\min}}$) to light ($R_{T_{\max}}$).

of high gradients (i.e. at the root and tips of the wing) acts to improve these first-order approximations.

7.1.2 Effect of Blending Length

Unlike the joint length parameter, δ/c, there is not an effective means of isolating the blending length, $\Delta\bar{z}$, for observation. For wings with sweep, the locus of aerodynamic centers is not at all points perpendicular to the freestream, resulting in the need for a finite joint length, as discussed in Chapter 2. Therefore, the cases of swept wings used to test the sensitivity of the general implementation's convergence to $\Delta\bar{z}$ will also be influenced by any effects from δ/c. Fortunately, based on the results of the previous section, the effects of joint length will be small for a large enough value of δ/c.

The parameters varied to study the effects of the blending length are shown in Table 7.2. Again, for each case, the RMS and lift coefficient convergence rates are recorded, and each

Table 7.2: Summary of the parameters varied to observe the effect of blending length on convergence.

Parameter	Min. Value	Max. Value	Std. Value
$\Delta\bar{z}$	0.01	2	—
Λ	$-30°$	$60°$	$30°$
δ/c	0.1	1	0.15
β	$0°$	$20°$	$5°$
α	$-5°$	$10°$	$5°$
R_A	2	24	8
R_T	0.1	1	1

parameter is sampled at 10 uniformly spaced values along its feasible range, while all others are held at a "standard" value. The results are shown in Figs. 7.5–7.10, relative to blending length, $\Delta\bar{z}$, as defined in Eq. (2.16). Again, the convergence rates are calculated both with a uniform node spacing (dashed lines) and a cosine-clustered node spacing (solid lines) [19].

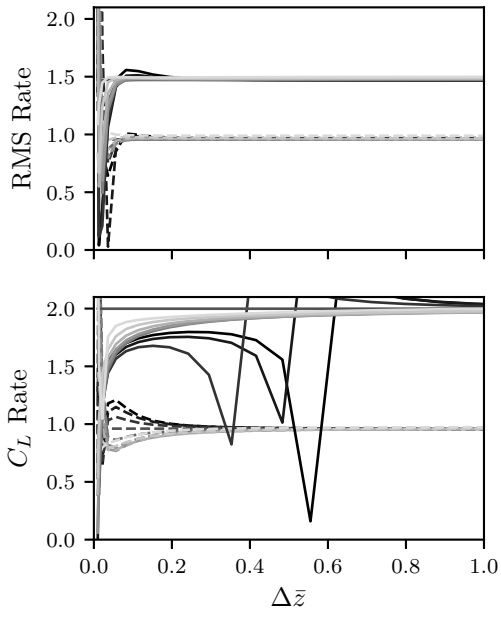

Fig. 7.5: The sensitivity of the implementation's convergence rate to blending length and sweep, dark ($\Lambda_{\min}$) to light ($\Lambda_{\max}$).

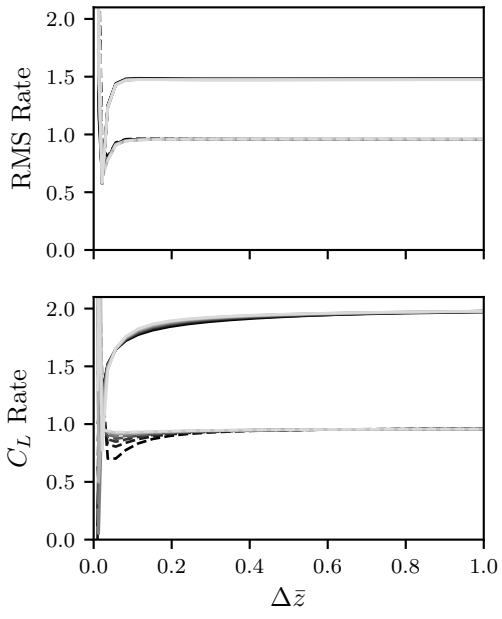

Fig. 7.6: The sensitivity of the implementation's convergence rate to blending length and joint length, dark ($\delta_{\min}/c$) to light ($\delta_{\max}/c$).

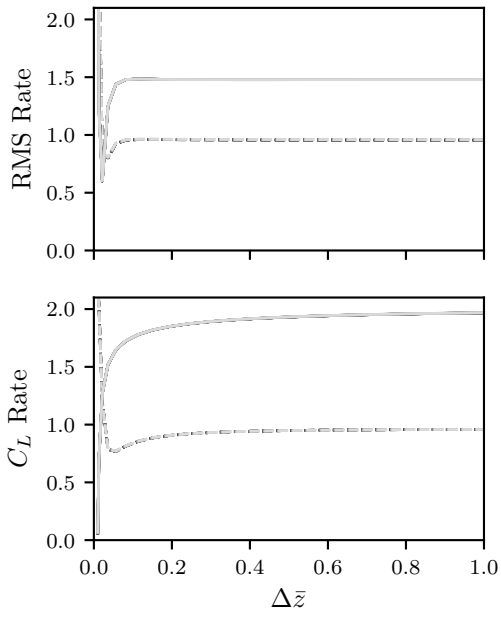

Fig. 7.7: The sensitivity of the implementation's convergence rate to blending length and side-slip angle, dark ($\beta_{\min}$) to light ($\beta_{\max}$).

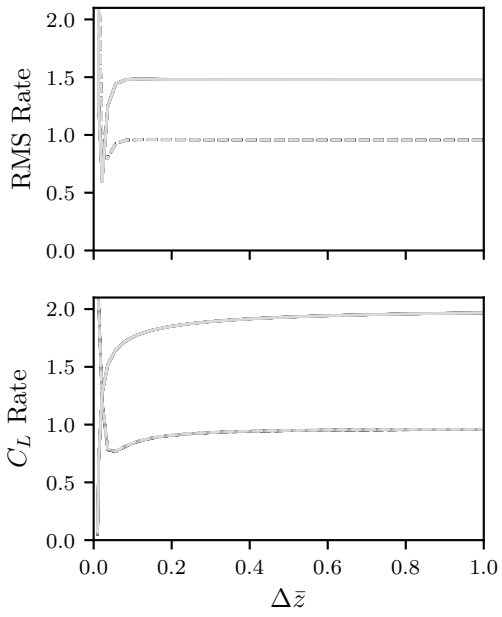

Fig. 7.8: The sensitivity of the implementation's convergence rate to blending length and angle of attack, dark ($\alpha_{\min}$) to light ($\alpha_{\max}$).

The results in Figs. 7.5–7.10 show similar behavior as that observed in the joint-length sensitivity study. Namely, for blending lengths above a given threshold—in this case approximately 0.25—the convergence of the general implementation of lifting-line theory remains essentially unaffected. The one notable exception is observed in the plot of the C_L convergence in Fig. 7.5, where the three negative sweep angles exhibit asymptotic behavior at approximately $\Delta\bar{z} = 0.4, 0.5$, and 0.6, respectively, when a cosine-clustered node distribution is used. This is due to the fact that, for wings with negative sweep, small blending lengths result in over-predictions of C_L at low node counts, before converging as the node count increases. Conversely, large blending lengths result in under-predictions of C_L at low node counts. Therefore, there exists a blending length for which the prediction of C_L is virtually independent of node-count, producing an erratic prediction from the Richardson extrapolation. The end result is the poor behavior observed in the lift-coefficient convergence rate in Fig. 7.5, for wings with negative sweep. However, from the RMS convergence

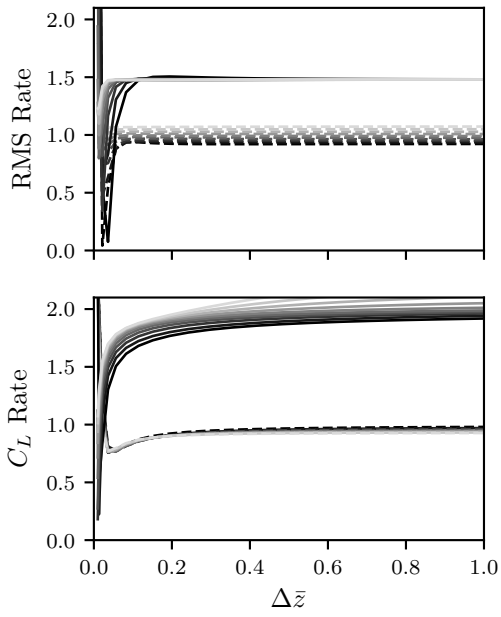

Fig. 7.9: The sensitivity of the implementation's convergence rate to blending length and aspect ratio, dark ($R_{A_{\min}}$) to light ($R_{A_{\max}}$).

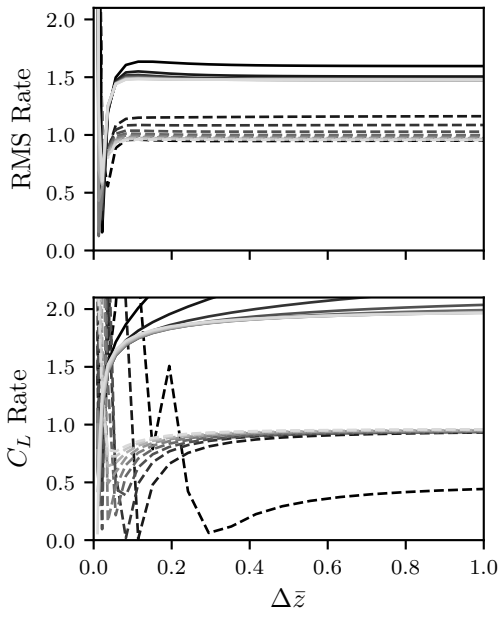

Fig. 7.10: The sensitivity of the implementation's convergence rate to blending length and taper ratio, dark ($R_{T_{\min}}$) to light ($R_{T_{\max}}$).

rates of those same cases, it is clear that the general implementation of lifting-line theory is still convergent.

The convergence rates for both the RMS and lift coefficient again demonstrate first-order convergence when using a uniform node distribution, with little to no variation with respect to any of the varied parameters, save for the case of $R_T = 0.1$ in the C_L convergence in Fig. 7.10, where the exhibited convergence rate is much lower than one. The use of a clustered node distribution increases the RMS and C_L convergence rates to about 1.5 and two, respectively. It is interesting to note that, for the cases of low aspect ratios in Fig. 7.10, the C_L convergence exceeds two when a clustered node distribution is used.

7.1.3 Comparison to Other Implementations

Having evaluated the convergence properties of the general implementation of lifting-line theory for a wide range of wing geometries, it is seen that joint-length values should

be chosen such that $\delta/c \gtrsim 0.15$, and blending lengths be chosen such that $\Delta\bar{z} \gtrsim 0.25$. Figure 7.11 shows the convergence rate of the total lift coefficient, C_L, as well as the convergence rate of the RMS change in circulation distribution for the case of the wing used in Weber and Brebner's experiment [20], for a range of δ/c and $\Delta\bar{z}$, using a clustered node distribution. As a comparison, the convergence of the finite-core implementation is also shown, for a range of core radii, r_c/c, using the same clustered node distribution. It can be seen in Fig. 7.11 that the convergence of the finite-core implementation behaves, with respect to changes in r_c/c, similar to the general implementation of lifting-line theory, with respect to changes in δ/c and $\Delta\bar{z}$. After a threshold value of $r_c/c \approx 0.2$, the convergence remains unaffected. The difference between the two methods, however, is seen to be the rate of convergence itself. It appears that the finite-core implementation of lifting-line does not benefit from node-clustering as does the general implementation. It demonstrates a C_L convergence of one, and an RMS convergence of 0.5.

As was done in the analysis of Phillips' implementation in Fig. (1.5), the general lifting-line theory implementation is used to predict the circulation distribution of Weber's wing, using a range of node counts, in order to demonstrate its numerical convergence. The results of this study are shown in Figure 7.12. In contrast to the non-convergent circulation distribution shown in Fig. 1.5, the results from the general implementation of lifting-line in Fig. 7.12 indicate a grid-convergent solution. This difference in grid-convergence demonstrates one advantage that the addition of jointed trailing vortices and conditional concavity provides to the general lifting-line theory implementation, as compared to Phillips' original numerical implementation. Similar studies, performed for Weissinger's method and the finite-core implementation, are shown in Figs. 7.13 and 7.14.

Table 7.3 shows a comparison of the convergence rates of the five numerical methods considered in this work. Recall that an algorithm is considered to be adequately convergent if its rate of convergence is at least one, while a convergence rate around two or above is preferable [21]. As observed before, Phillips' method is not convergent, which is exhibited by the sub-one C_L and RMS convergence rates. The finite-core method is approximately first-

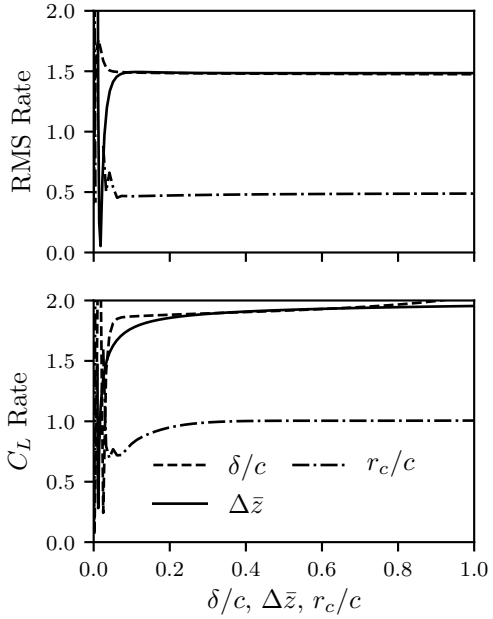

Fig. 7.11: The sensitivity of the general lifting-line implementation's convergence rate to blending length (solid) and joint length (dashed), as compared to a finite-core vortex model (dotted).

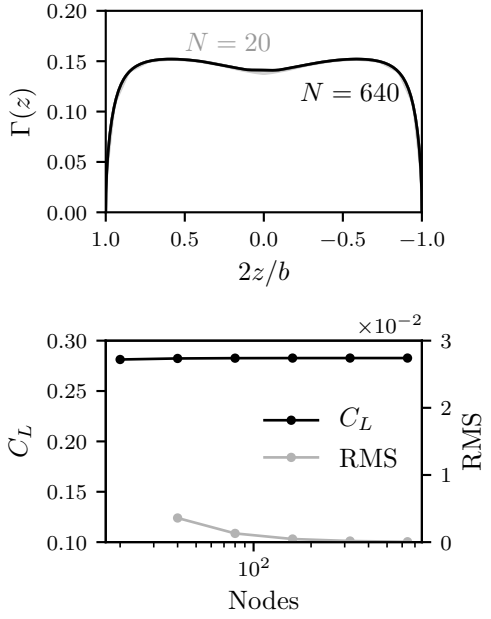

Fig. 7.12: The circulation distribution predicted by the general implementation, for several node counts ($\Delta\bar{z} = 0.25$ and $\delta/c = 0.15$) (top). The RMS change in the circulation distributions and the total lift coefficient as a function of the node count (bottom).

Table 7.3: Summary of convergence rates.

Implementation	C_L Rate	RMS Rate
Phillips'	0.129	0.635
Finite-Core	0.902	0.469
General	1.875	1.487
Weissinger's	2.252	1.480
PAN AIR	3.405	—

order convergent in its predicted lift coefficient, but does not demonstrate convergence in its prediction of the circulation distribution, as observed in Fig. 7.14. Both the general lifting-line theory implementation and Weissinger's method demonstrate approximately second-order convergence in their predictions of C_L, and show a convergence rate of approximately 1.5 in the circulation distribution. PAN AIR takes advantage of higher-order approximations than those used in lifting-line theory, resulting in a convergence rate of almost 3.5 for C_L.

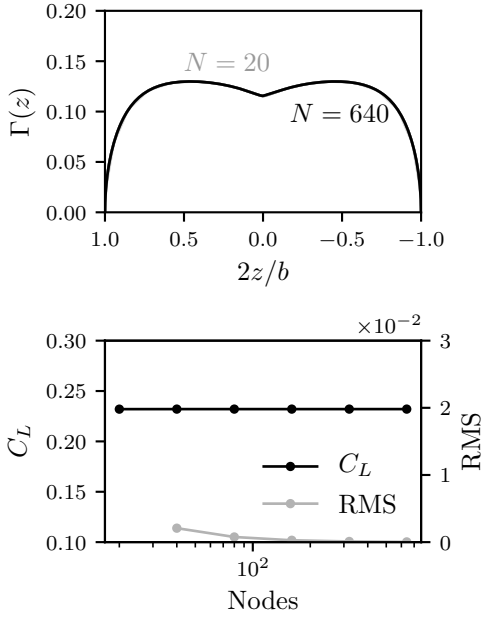

Fig. 7.13: The circulation distribution predicted by Weissinger's method, for several node counts (top). The RMS change in the circulation distributions and the total lift coefficient as a function of the node count (bottom).

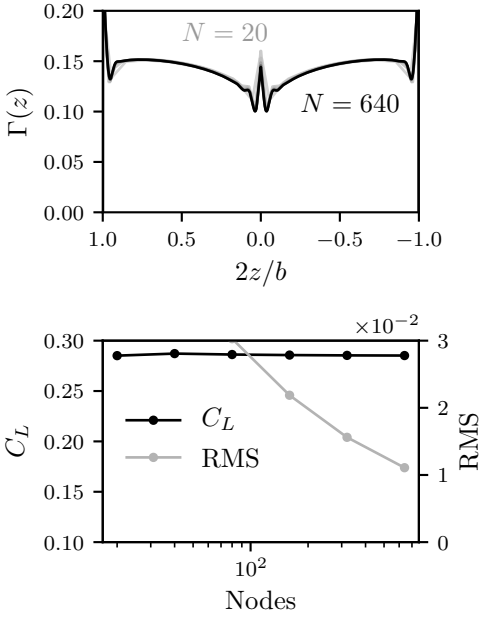

Fig. 7.14: The circulation distribution predicted by the finite-core implementation, for several node counts ($r_c/c = 0.15$) (top). The RMS change in the circulation distributions and the total lift coefficient as a function of the node count (bottom).

7.2 Accuracy

Having established the convergence of the general lifting-line theory implementation, an analysis is made of its accuracy. First, based on the discussion in Chapter 5, recall the four methods of predicting the section properties in the general implementation of lifting-line theory. The section-lift properties of the wing can be approximated using thin-airfoil theory, as described in Eq. (5.6). More generally, the section properties of the wing can be approximated using Eq. (5.35), where the effective circulation, Γ_Λ, is calculated directly using a vortex panel method (VPM). Because the use of the vortex panel method directly is computationally expensive, Eq. (5.44) can be used, with one of two methods for calculating the effective lift slope, $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$, and effective zero-lift angle of attack, α_{L0_Λ}. First, $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$ and α_{L0_Λ} can be found be fitting a curve to data from a vortex panel method for each section. Second, the curve fit approximations defined in Eqs. (5.61) and (5.63) can be

used to approximate $\tilde{C}_{L_\Lambda,\alpha_\Lambda}$ and α_{L0_Λ}. The computation time required to analyze Weber and Brebner's wing are given in Table 7.4, and the error of each section-properties model, relative to the results obtained by evaluating the vortex panel method directly, are shown in Fig. 7.15.

Table 7.4: Example of computational cost for section property models ($N = 160$).

	Thin-Airfoil	VPM	VPM Fit	Approx.
Set-Up Time (s)	0.003	0.886	0.915	0.003
Solve Time (s)	0.193	14.356	0.040	0.040

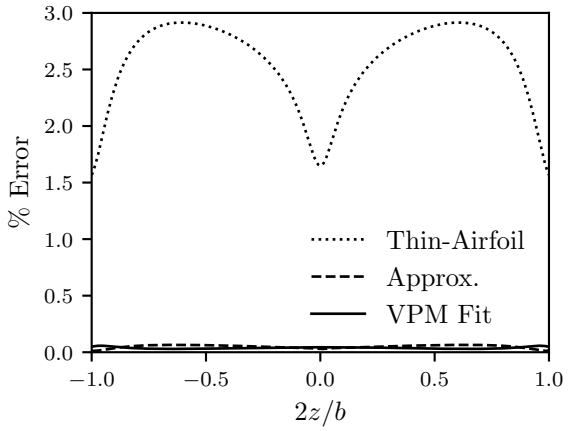

Fig. 7.15: Comparison of the section property models discussed in Chapter 5.

The example results in Table 7.4 show the high computational cost of evaluating vortex panel method directly, compared to the other three approximations. Furthermore, considering the low relative error of the vortex panel fit and approximation shown in Fig. 7.15, there does not appear to be much lost by approximating the vortex panel method results using Eq. (5.44). Recall that the error of these predictions is shown in Fig. 5.19 for infinite wings. The error in Fig. 7.15 is similar in form to the errors displayed in Fig. 5.19. The approximation made using thin-airfoil theory (i.e. Eq. (5.6)) demonstrates about 1.5% error at the root and tips, where the effective sweep angle is low, and about 3% at the mid-span where

the effective sweep is highest. The other two approximations agree very closely with the results vortex panel method. It should be noted, however, that the approximations defined in Eqs. (5.61) and (5.63) are most accurate for NACA 4-digit airfoils, such as those used in this study. For airfoils outside the NACA 4-digit family, the results of the approximation may be less stunning. For the analyses herein, a curve fit of vortex panel method data is used to predict the section properties for each wing section.

The section lift distributions predicted by each of the methods described at the beginning of the chapter are shown in Fig. 7.16, normalized by the total lift coefficient, for Weber and Brebner's wing at an angle of attack of 4.2°. It is seen in Fig. 7.16 that the general lifting-line theory implementation, Weissinger's method, and PAN AIR predict distributions similar in shape to the experimental results. The lift distribution predicted by Phillips' implementation and the finite-core implementation, on the other hand, do not accurately predict the lift distribution, under-predicting the lift at the center and over-predicting the lift at the tip.

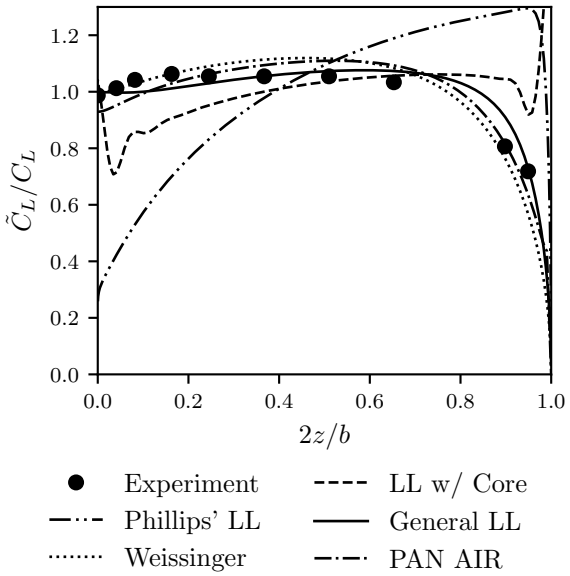

Fig. 7.16: The lift distribution, as predicted by the general lifting-line implementation, Phillips' implementation, a finite-core implementation, the panel method PAN AIR, and experimental data by Weber and Brebner, at an angle of attack of 4.2°.

The total wing lift coefficients predicted by the six methods are shown in Fig. 7.17, as functions of the section-lift coefficient, $\tilde{C}_{L_\infty}$, of a straight infinite wing with the same airfoil, at the same angle of attack. The values of $\tilde{C}_{L_\infty}$ for four of the inviscid numerical prediction methods were found using a 2-D vortex panel method [5], resulting in a section lift slope of 6.907. Because Weissinger's method is based on flat-plate theory, the section lift slope of 2π is used to normalize its results. Although Weber doesn't report the 2-D lift slope of the airfoil used in his work, experimental results from other sources, at the same Reynolds number (1.68×10^6), predict a viscous section lift slope of 5.935 [53]. Again, the general lifting-line implementation, Weissinger's method, and PAN AIR match the results of the experimental work. The finite-core implementation also demonstrates good agreement with the experimentally-predicted total lift coefficient, despite its inaccuracy in predicting the lift distribution. The general lifting-line implementation and finite-core implementation show better agreement with the experimental results at lower lift coefficients, and Weissinger's method and PAN AIR more closely match the lift coefficients of the experiments at high lift coefficients. Conversely, Phillips' implementation drastically under-predicts the lift at

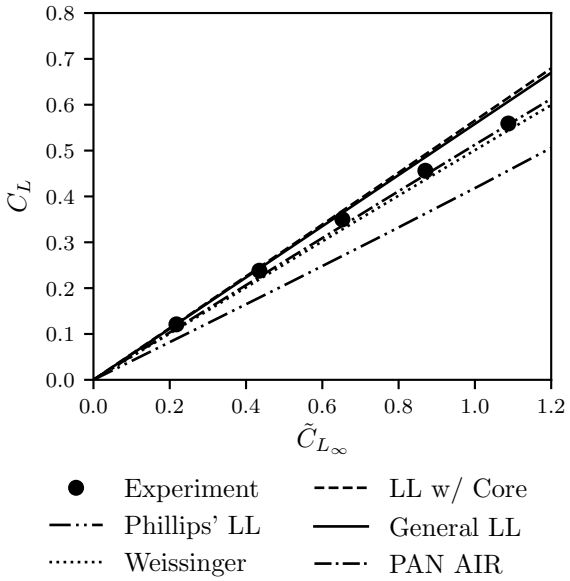

Fig. 7.17: The total lift coefficient, as predicted by the general lifting-line implementation, Phillips' implementation, a finite-core implementation, the panel method PAN AIR, and experimental data by Weber and Brebner.

all lift coefficients.

The predicted drag coefficients are shown in Fig. 7.18. The experimental results are gathered using pressure taps along the surface of the wing, and do not account for friction drag. Nevertheless, the experimental results are subject to viscous effects that increase the calculated drag, such as chordwise and spanwise boundary layer growth [20]. The inviscid numerical methods, on the other hand, predict only induced drag and are not influenced by viscous drag effects. Therefore, it is expected that the inviscid numerical methods should predict lower drag values than those obtained from experiment. The general implementation of lifting-line theory, the finite-core implementation, Weissinger's method, and PAN AIR meet this expectation. The under-prediction is small at low lift coefficients and increases with increasing lift, where the effects of viscosity are increasingly dominant. Furthermore, these methods demonstrate good agreement with the induced drag approximation made by

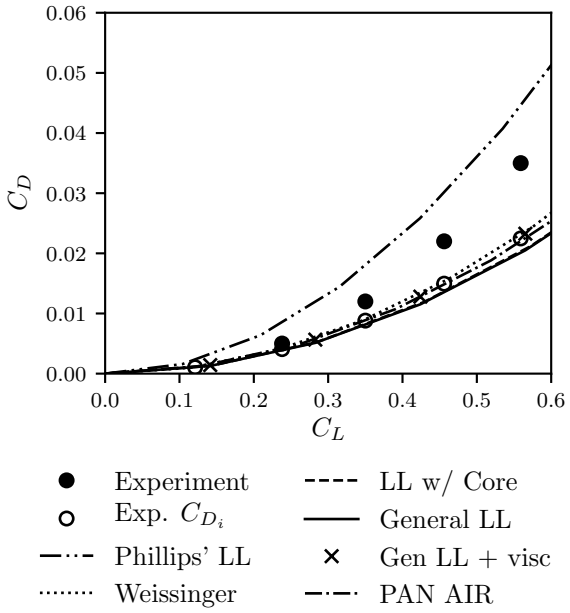

Fig. 7.18: The total drag coefficient, as predicted by the general lifting-line implementation, Phillips' implementation, a finite-core vortex implementation, the panel method PAN AIR, and experimental data by Weber and Brebner.

Weber and Brebner [20] using

$$C_{D_i} = \frac{C_L^2}{\pi e R_A} \tag{7.1}$$

where e was found by Weber and Brebner to be approximately 0.885 for this wing and angle of attack. On the other hand, there is an over-prediction of drag by Phillips' implementation of lifting-line theory. The high drag prediction by Phillips' method is likely because the implementation predicts a circulation distribution with high gradients, $d\Gamma/dz$, along the majority of the wing, increasing the induced drag by increasing the overall strength of the trailing vortex sheet.

Using the viscous correction defined in Eq. (6.46), with viscous drag predictions obtained using XFOIL [49], a prediction of the viscous effects to the pressure drag is also shown in Fig. 7.18. The prediction for the viscous pressure drag is seen to still under-predict the drag calculated from experimental data. This under-prediction hints at the large role spanwise boundary layer growth plays in the resulting drag force. Because the methods described herein in the general implementation of lifting-line theory assume section flow at each point along the wing, the implementation is not equipped, as presented here, to predict such spanwise effects. However, if section properties were to be obtained that take into account the spanwise boundary layer growth, the general implementation of lifting-line theory would be expected to more accurately predict the resulting drag [32].

The comparisons between the general lifting-line theory implementation, Phillips' implementation, a finite-core modification to Phillips' implementation, Weissinger's method, PAN AIR, and Weber and Brebner's experimental results serve to validate the accuracy of the general lifting-line theory implementation. The lift distribution, total lift coefficients, and total drag coefficients predicted by general lifting-line show good agreement with those of PAN AIR and the experimental results, and show the rectification of the issues faced when applying Phillips' method to wings with sweep. Because the NACA 0012 airfoil used in this test case is closely approximated by a flat plate, Weissinger's method also replicates the experimental results and the predictions from PAN AIR. It is anticipated that this level of agreement between Weissinger's method and PAN AIR would decrease if camber were

added to the wing. These comparisons also illustrate the inconsistent convergence of the finite-core implementation. The finite-core method demonstrates good agreement for the total lift and drag predictions, but predicts an inaccurate lift distribution.

It is worth noting that the results of the lifting-line theory implementation were obtained at a computational cost four orders of magnitude (10^{-4}) less than that required by PAN AIR, and as little as 10^{-6} of the cost required by CFD software [19].

7.3 Sensitivity to Closure Parameters

The predictions made in Figs. 7.16–7.18 by the general implementation of lifting-line theory depend on the closure parameters $\Delta\bar{z}$ and δ/c. The effect that those parameters have on the predictions of the spanwise distribution of lift shown in Fig. 7.16 is demonstrated in Figs. 7.19 and 7.20. As a comparison, the effect of r_c/c on the results of the finite-core implementation is shown in Fig. 7.21. As the joint length, δ/c, increases, the predicted lift distribution increases slightly across the inboard section of the wing and decreases near the tip. Similarly, an increase in the blending length, $\Delta\bar{z}$, increases the predicted lift at the center of the wing and decreases the lift predicted towards the tip, though the changes at the center are larger than those observed for changes in the joint length. Notice that, as $\Delta\bar{z}$ increases to one, the change in the predicted lift distribution with respect to $\Delta\bar{z}$ decreases,

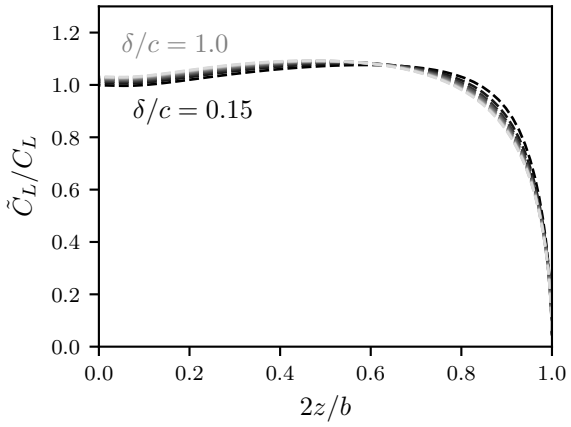

Fig. 7.19: Lift distributions predicted by the general lifting-line implementation, with changing joint length, δ/c.

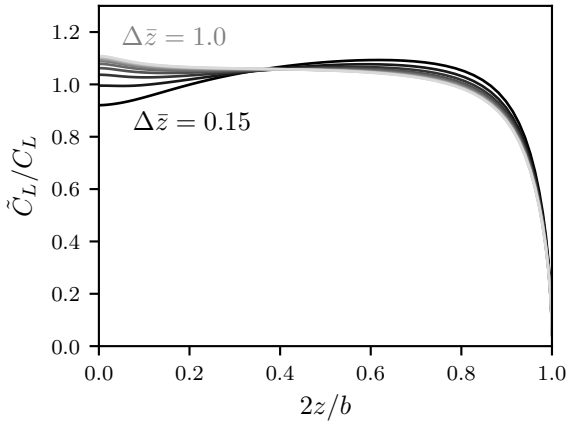

Fig. 7.20: Lift distributions predicted by the general lifting-line implementation, with changing blending length, $\Delta\bar{z}$.

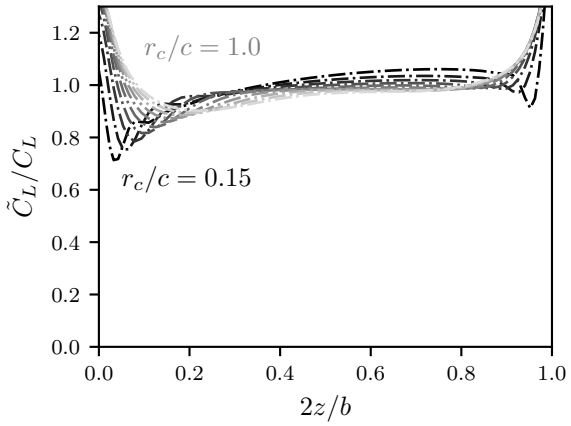

Fig. 7.21: Lift distributions predicted by the finite-core lifting-line implementation, with changing core radius, r_c/c.

suggesting that increasing $\Delta\bar{z}$ above a certain value will no longer affect the predicted lift distribution. The trend observed in Fig. 7.21 for an increase of core radius, r_c/c, is somewhat different than those observed for δ/c and $\Delta\bar{z}$. As the core radius increases, the predicted lift increases at the root and tip, while decreasing in the midspan. The predictions of the lift distribution shown in Fig. 7.21 also appear to change less as the the core radius increases, though to a lesser extent than observed in Fig. 7.20.

The sensitivity of the predictions made by the general lifting-line theory and finite-core model shown in Figs. 7.19–7.21 can be quantified using extrapolated values for the lift coefficient (i.e. $N = \infty$) to calculate the gradient of the total lift coefficient, ∇C_L, with respect to δ/c, $\Delta\bar{z}$, and r_c/c, shown in Fig. 7.22. Over the range of values shown, the prediction of the lift coefficient is shown to be more sensitive to the radius used in the finite-core vortex model than to the blending length and joint length used in the general lifting-line implementation. For the values used in Figs. 7.16–7.18, $\delta/c = 0.15$, $\Delta\bar{z} = 0.25$, and $r/c = 0.15$, the lift coefficient sensitivity, ∇C_L, is shown to be approximately 0.023, 0.062, and 0.165, respectively. These values can be interpreted to suggest that changes in δ/c, $\Delta\bar{z}$, and r/c on the order of 0.1 will lead to changes in the predicted total lift coefficient of approximately 0.0023, 0.0062, and 0.017, respectively.

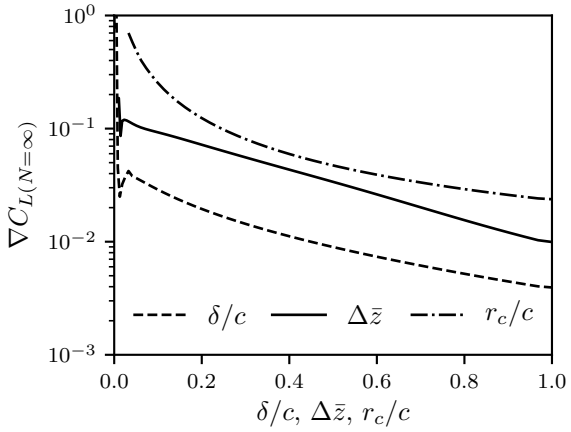

Fig. 7.22: The sensitivity of the general lifting-line implementation's results to blending length and joint length, as compared to a finite-core vortex model.

The values of δ/c and $\Delta\bar{z}$ used to obtain the results shown in Figs. 7.16–7.18 were chosen solely on the criteria of convergence, with the purpose of demonstrating the capability of the general implementation for a convergent case with a negligible amount of adjustment. In practice, these values could be adjusted to further improve the agreement between the general lifting-line theory implementation and a specific case-study. Figure 7.23 shows the accuracy of the general lifting-line theory implementation for a range of values for δ/c

and $\Delta\bar{z}$, measured as the RMS difference between the predicted lift distribution and the distribution found in Weber and Brebner's experiment [20]. The dotted lines in Fig. 7.23 mark the convergence threshold values for δ/c and $\Delta\bar{z}$. The values used in this work, $\delta/c = 0.15$ and $\Delta\bar{z} = 0.25$, result in an RMS error of approximately 0.04. The results in Fig. 7.23 show that this error could be reduced below 0.032 if the joint length were increased to approximately 0.5.

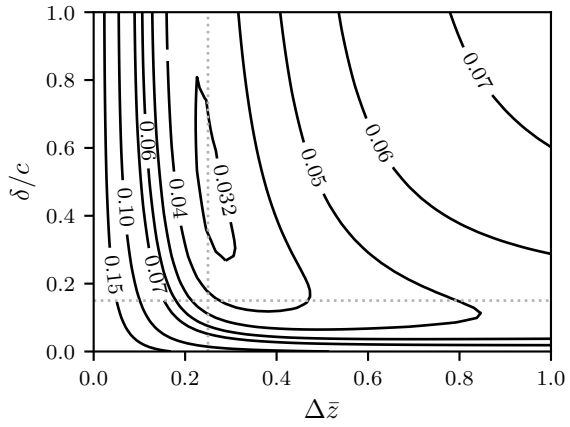

Fig. 7.23: The RMS difference in the lift distribution predicted by the general lifting-line implementation and experimental data, as a function of blending length, $\Delta\bar{z}$, and joint length, δ/c. The dotted gray lines represent the threshold values of $\Delta\bar{z}$ and δ/c.

The sensitivity analyses shown in Figs. 7.22 and 7.23 were performed on the results for the wing used in Weber and Brebner's experiment [20]. Now, consider the range of wings described by Tables 7.1 and 7.2. The gradient of the lift coefficient, ∇C_L, for the straight wings in sideslip described by Table 7.1, is shown in Figs. 7.24–7.27. Similarly, Figs. 7.28–7.33 show ∇C_L for the swept wings described by Table 7.2.

The lift coefficient gradient for the straight wings in sideslip, shown in Figs. 7.24–7.27, remains under 0.07 for all joint lengths greater than 0.15. The lift coefficient gradient for the swept wings, shown in Figs. 7.28–7.33, remains under 0.1 for all blending lengths greater than 0.25, save for the case of the highest angle of attack in Fig. 7.31 where the highest value of ∇C_L is 0.12. This means that, for the changes of δ/c or $\Delta\bar{z}$ on the order of 0.1,

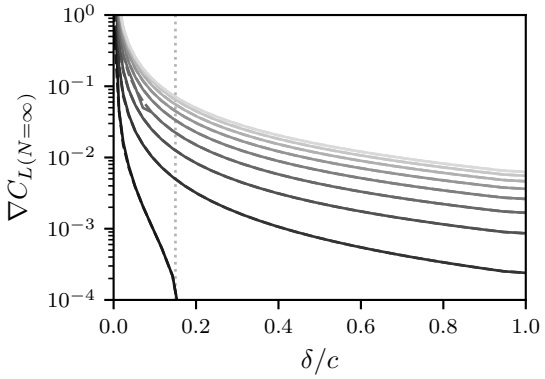

Fig. 7.24: Lift coefficient sensitivity to joint length and side-slip angle, dark ($\beta_{\min}$) to light ($\beta_{\max}$).

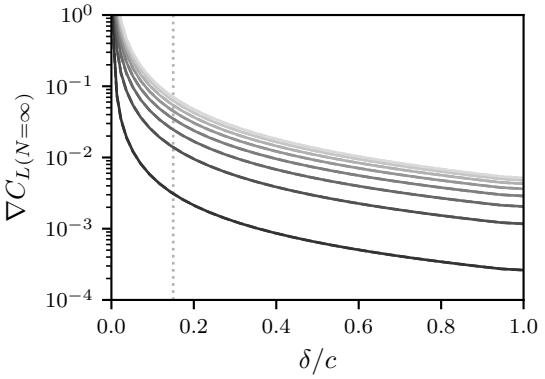

Fig. 7.25: Lift coefficient sensitivity to joint length and angle of attack, dark ($\alpha_{\min}$) to light ($\alpha_{\max}$).

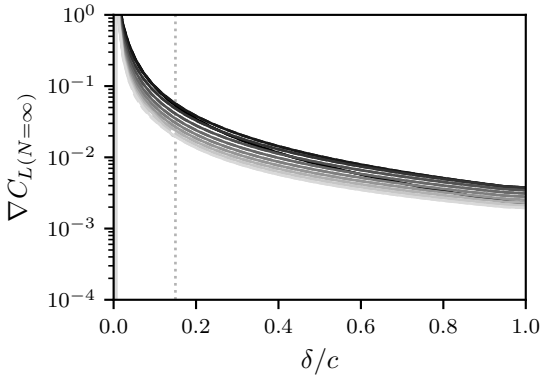

Fig. 7.26: Lift coefficient sensitivity to joint length and aspect ratio, dark ($R_{A_{\min}}$) to light ($R_{A_{\max}}$).

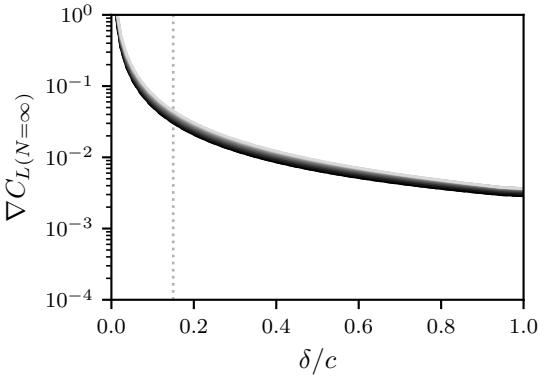

Fig. 7.27: Lift coefficient sensitivity to joint length and taper ratio, dark ($R_{T_{\min}}$) to light ($R_{T_{\max}}$).

the predicted lift coefficient is expected to change by less than 0.01 for all convergent cases.

The results shown in Figs. 7.24–7.33 include values calculated both using a uniform node distribution (dashed lines) and using cosine-clustered nodes (solid lines). Because extrapolated values are used to calculate ∇C_L, in almost all cases the results are the same using both node spacings, indicating that the results converge to the same value, though at different rates. The only exception can be seen in Fig. 7.33 for low taper ratios (dark lines). Recall, from Fig. 7.10, that the the results for low taper ratios calculated using a uniform node distribution exhibited poor convergence for values of $\Delta\bar{z}$ less than about 0.4.

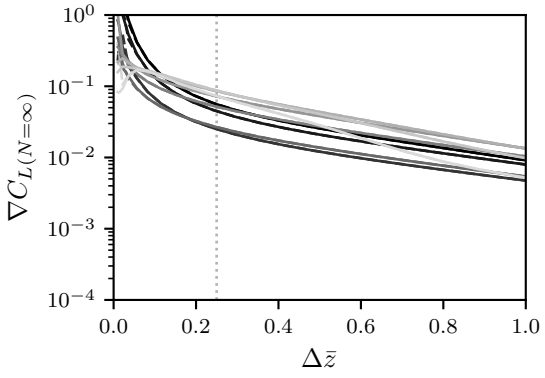

Fig. 7.28: Lift coefficient sensitivity to blending length and sweep, dark ($\Lambda_{\min}$) to light ($\Lambda_{\max}$).

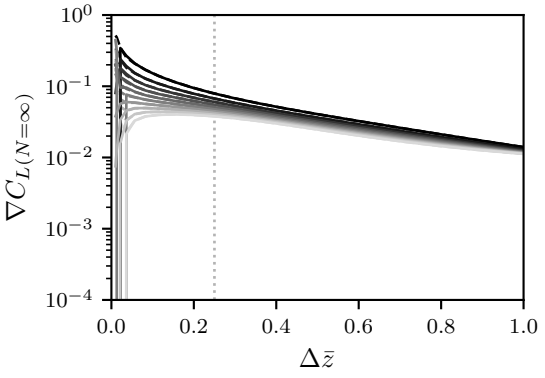

Fig. 7.29: Lift coefficient sensitivity to blending length and joint length, dark ($\delta_{\min}/c$) to light ($\delta_{\max}/c$).

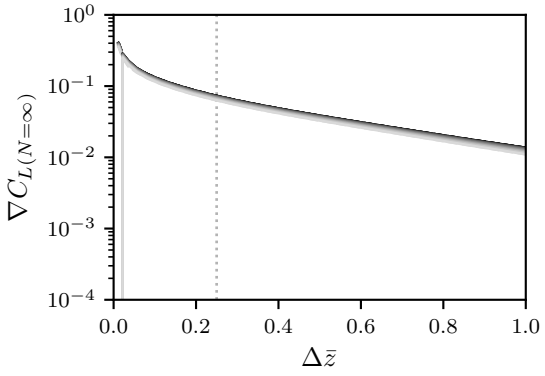

Fig. 7.30: Lift coefficient sensitivity to blending length and side-slip angle, dark ($\beta_{\min}$) to light ($\beta_{\max}$).

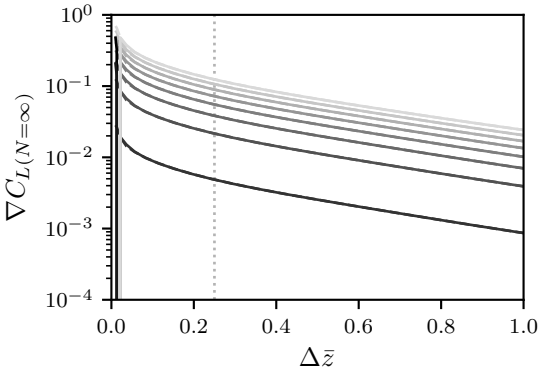

Fig. 7.31: Lift coefficient sensitivity to blending length and angle of attack, dark ($\alpha_{\min}$) to light ($\alpha_{\max}$).

Therefore, because they converge to different values (the uniform node spacing resulting in an incorrect value) the extrapolated values differ between the results of the clustered node-spacing and the uniform node distribution.

While, in the majority of cases, the lift coefficient gradient gradually decreases as the joint length increases, in the study of the effect of sideslip shown in Fig 7.24, ∇C_L rapidly approaches zero around $\delta/c = 0.15$, for the case of zero sideslip (black line). For values of δ/c less than 0.15, the finite lift coefficient gradient results from the fact that, even though there is no sideslip, the finite and semi-infinite portions of the trailing vortices are not parallel due to the angle of attack of the flow (recall from Table 7.1 that the standard value

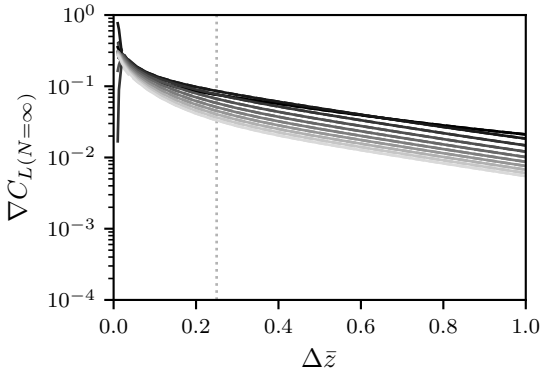

Fig. 7.32: Lift coefficient sensitivity to blending length and aspect ratio, dark ($R_{A_{\min}}$) to light ($R_{A_{\max}}$).

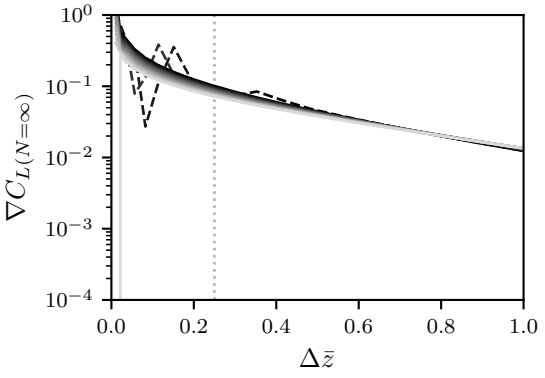

Fig. 7.33: Lift coefficient sensitivity to blending length and taper ratio, dark ($R_{T_{\min}}$) to light ($R_{T_{\max}}$).

for α is 5°). From the results in Fig. 7.24 it appears that once the joint length exceeds fifteen percent of the chord length, the influence of that angle becomes negligible.

The sensitivity studies shown in Figs. 7.22 and 7.33 suggest that adjusting the joint length, δ/c, and blending length, $\Delta\bar{z}$, provides relatively small variations in the calculated results. While these small variations may be helpful when trying to adjust the results of the general implementation of lifting-line theory to match known data for a particular wing, the fact that the variations are small suggests that good predictions can be made without the need of adjustment, as was demonstrated with the analysis of Weber and Brebner's wing shown in Fig. 7.23. If needed, larger adjustment of the results can be achieved through the modification of the airfoil section properties used in the computation (e.g. to account for viscous effects, etc.).

In the analyses performed in this chapter, the general implementation of lifting-line theory is shown to be approximately second-order convergent and demonstrate a level of accuracy consistent with widely-used potential-flow methods. Furthermore, it is shown that its convergence and accuracy are relatively insensitive to the joint and blending lengths used in the implementation, for values of δ/c and $\Delta\bar{z}$ above a certain threshold. These analyses confirm that the general implementation of lifting-line theory presented in this work is applicable to the prediction of the aerodynamic properties of swept wings and wings in sideslip, while avoiding the drawbacks of the other implementations discussed herein. In

particular, the general implementation of lifting-line theory permits the use of arbitrary models for section properties, is relatively insensitive to model closure parameters, and is numerically convergent.

CHAPTER 8

CONCLUSION AND FUTURE WORK

Prandtl's classical implementation of lifting-line theory has been used widely, and provides valuable intuition into the aerodynamic behavior of finite wings. However, in Prandtl's classical form, Eq. (1.9), lifting-line theory is restricted to straight wings in flows without sideslip. Similarly, the numerical implementation of lifting-line theory developed by Phillips, Eq. (1.15), suffers from an inability to grid-converge for swept wings or wings in sideslip, as seen in Fig. 1.5, though it removes the need for integration in the calculation of induced velocities. The limitations of these two implementations of lifting-line theory have been carefully considered in the construction of a more-general implementation that is capable of predicting the lift of wings with sweep and wings in sideslip.

The limitation of the previous lifting-line theory implementations is rooted in the model used to predict induced velocities. As shown in Eq. (2.9), the total velocity induced by the vortex filament, at a point along its length, is finite only if the second derivative of the vortex filament, which lies along the locus of aerodynamic centers, is zero. Similarly, Eq. (2.29) implies that the trailing vortices must be perpendicular to the locus of aerodynamic centers in the neighborhood of the point of interest for the influence of the trailing vortex sheet to remain finite. Prandtl's classical implementation of lifting-line theory satisfies these conditions because it is restricted to cases of straight wings with zero sideslip. In Weissinger's method, the infinite induced velocities are avoided by calculating the vortex influences off of the bound vortex filament, but in doing so, the section properties of the wing are restricted to those of a flat plate. A finite-core vortex model can be used, but it results in slower grid convergence, an inaccurate prediction of the lift distribution, and sensitivity to the selected core size. It is shown in Sections 2.1.1 and 2.2.1 that Phillips' numerical implementation is subject to the same restrictions on vortex geometry as Prandtl's implementation and therefore does not grid-convergence for cases in which they are not satisfied (i.e. non-straight

wings or wings in sideslip).

To satisfy the conditions on vortex geometry for non-straight wings or wings in sideslip, a model is presented in Eq. (2.13) in which the second derivative of the locus of aerodynamic centers is forced to zero at a given control point and blended with the original curve to minimize the deviation at the other points along the span of the wing. The result is an effective locus of aerodynamic centers for each point along the bound vortex filament. Furthermore, each trailing vortex is jointed, as depicted in Fig. 2.6, such that there is a finite segment of the trailing vortex perpendicular to the locus of aerodynamic centers and a semi-infinite portion aligned with the freestream. The modifications to the bound vortex filament and trailing vortex sheet provide the flexibility needed to implement lifting-line theory for wings with sweep or in sideslip.

The influence of a parabolic vortex segment is also considered. The resulting closed-form solution demonstrates an elegant relation between the induced velocity field and classic algebraic geometry. However, due to the computational cost of its application, it is ultimately not applied to the general implementation of lifting-line theory.

In addition to the considerations given to the evaluation of the induced velocity, consideration is given to modeling the locus of aerodynamic centers of swept wings and the section aerodynamic properties of such wings. Thin-airfoil theory predicts a reduction of section lift when sweep is applied to an infinite wing, described by Eq. (5.6). It is found that, as sweep is introduced to an infinite wing, the effective freestream velocity, angle of attack, and airfoil geometry change. By characterizing those changes, a prediction for the change in total section circulation is found, described in Eq. (5.35). Using a two-dimensional vortex panel method, good agreement is seen when compared to the section-lift slope calculated in CFD simulations, as observed in Figs. 5.18 and 5.19. The approximation defined in Eq. (5.44) is also shown to be accurate. The findings from the infinite wing are applied to each section of a finite wing with sweep according to the section's effective sweep angle, defined by the locus of aerodynamic centers.

Using the effective loci of aerodynamic centers, jointed trailing vortices, and swept section properties, the general implementation of lifting-line theory is applied in a similar manner as Phillips' numerical algorithm. The resulting application of lifting-line theory is able to predict the circulation distribution of wings with sweep and wings in sideslip, with approximately second-order convergence, demonstrated in Figs. 7.1–7.12, so long as the joint length and blending length remain above the requisite threshold (i.e. $\delta/c \gtrsim 0.15$ and $\Delta\bar{z} \gtrsim 0.25$). Furthermore, this implementation of lifting-line theory is shown to be accurate when compared to experimental results, results from a high-order panel method, and traditional low-fidelity models in Figs 7.16–7.18.

The results of the general implementation of lifting-line theory used in the comparison against experimental results were obtained using largely arbitrary values for δ/c and $\Delta\bar{z}$, with the purpose of demonstrating the "out of the box" capability of the general implementation. In practice, these values could be adjusted, to further improve the agreement between the general lifting-line implementation and a specific case-study, as shown in Fig. 7.23. However, the sensitivity studies shown in Figs. 7.19–7.33 suggest that such tuning would only provide relatively small variations in the calculated results—less than 10^{-2} for changes in δ/c and $\Delta\bar{z}$ on the order of 10^{-1}.

The predictions from the general implementation of lifting-line theory were obtained at a computational cost four orders of magnitude (10^{-4}) less than that required by the high-order panel method PAN AIR, and as little as 10^{-6} of the cost required by CFD software. The computational savings demonstrated by the general implementation of lifting-line theory, in conjunction with its relatively high level of accuracy, show the important place that it holds, along with other low-fidelity methods, in the early stages of the aerodynamic design process, when many cases are required to explore a design space, prohibiting the use of high-cost methods. Additionally, the low computational cost of this method makes it a prime candidate for use in real-time applications, such as control algorithms or flight simulators. Using the general approach discussed in this work, these, and other, benefits of lifting-line theory can be applied to the analysis of wings with sweep and wings in sideslip.

Though this body of work presents a thorough approach to a general implementation of lifting-line theory, the research presented herein can be furthered in several aspects. First, the scope of this work is limited to planar wings. As such, many of the equations presented herein are confined to the x-z plane. These equations can readily be extended to three dimensions to encompass the cases of non-planar wings.

Second, the implementation of lifting-line theory presented in this work uses a discrete number of linear, constant-strength vortex segments to approximate a bound vortex filament, and a finite number of trailing vortices to approximate a continuous vortex sheet. These horseshoe vortices result in a first-order approximation of the continuous system, and could be replaced by a higher-order approximation. For example, the bound vortex filament could be approximated by a number of linear vortex segments whose strength varies linearly, and the continuous vortex sheet could similarly be approximated by a finite number of constant-strength jointed-vortex sheets, similar to the vortex sheets described in Sections 2.2.2 and 2.2.3. This type of higher-order approximation has been investigated as an extension of Phillips' implementation of lifting-line theory, but could be reworked to incorporate the advantages of the general implementation developed herein.

Third, further investigation is required to more-generally define the shape of the locus of aerodynamic centers for wings other than the planar wings of constant sweep considered in this work. In Chapter 4, Küchemann's approximation for the locus of aerodynamic centers is extended to wings of non-constant sweep. That extension, however, requires further analysis to determine its validity. Furthermore, the effect of dihedral on the shape of the locus of aerodynamic centers merits exploration before the general implementation is accurately applied to the analysis of non-planar wings. The generalized modeling of the locus of aerodynamic centers could perhaps be performed using an iterative application of the general implementation of lifting-line theory. Using an initial guess for the shape of the locus of aerodynamic centers, the general implementation could be used to obtain the aerodynamic information necessary to compute the locus of aerodynamic centers for the wing. Then, using the newly-computed locus, the requisite aerodynamic values would be

recomputed, and the process would repeat until the predicted shape of the locus of aerodynamic centers remains constant. If an accurate, convergent iteration could be achieved, the general implementation of lifting-line theory would no longer depend on approximations obtained by other means, and would be useful in providing rapid predictions that could be used to study the shape of the locus of aerodynamic centers for a large range of wing geometries.

Fourth, further analysis would extend the applicability of the swept-wing-section properties discussed in Chapter 5. If more airfoil data is obtained, empirical fits, similar to those in Eqs. (5.61), (5.63), (5.65), and (5.67), could be developed for families of airfoils other than the NACA 4-digit series. Additionaly, research into the spanwise viscous effects on swept wing sections properties could be included in the general implementation to provide better predictions of viscous drag than those observed in Fig. 7.18.

Finally, continued validation of the general implementation of lifting-line theory is required to further understand the accuracy and limitations of the method. In this work, the general implementation is compared against the experimental data obtained for one wing. Comparison against more experimental and higher-fidelity computational results for a variety of wing geometries and flow conditions will further evaluate the performance of the general implementation of lifting-line theory as a viable aerodynamic analysis tool. As with any new analysis method, the general implementation of lifting-line theory will be validated by the "test of time", through extended use in the aerodynamics community.

REFERENCES

[1] Prandtl, L., *Tragflügel Theorie*, Nachricten von der Gesellschaft der Wissenschaften zu Göttingen, Geschäeftliche Mitteilungen, Klasse, 1918, pp. 451–477.

[2] Prandtl, L., "Appplications of Modern Hydrodynamics to Aeronautics," Tech. Rep. TR-116, NACA, June 1921.

[3] Anderson, J. D., *Fundamentals of Aerodynamics*, chap. 1-5, McGraw-Hill, New York, 5th ed., 2011, pp. 1–486.

[4] Kuchemann, D., *The Aerodynamic Design of Aircraft*, Pergamon Press Ltd, Oxford, 1st ed., 1978.

[5] Phillips, W. F., *Mechanics of Flight*, chap. 1, John Wiley & Sons, Inc., New Jersey, 2nd ed., 2010, pp. 1–134.

[6] Karamcheti, K., *Principles of Ideal-Fluid Aerodynamics*, chap. 19, John Wiley & Sons, Inc., New York, 1966, pp. 548–554.

[7] Kutta, M. W., "Auftriebskräfte in Strömenden Flüssigkeiten," *Illustrierte Aeronautische Mitteilungen*, Vol. 6, 1902, pp. 133–135.

[8] Joukowski, N. E., "Sur les Tourbillons Adjionts," *Traraux de la Section Physique de la Societé Imperiale des Amis des Sciences Naturales*, Vol. 13, 1906, pp. 261–284.

[9] Ashenberg, J. and Weihs, D., "Curved Lifting-Line Theory for Thin Planar Wings," *Israel Journal of Technology*, Vol. 20, January 1982, pp. 160–165.

[10] Beyer, F., Matha, D., Sebastian, T., and Lackner, M., "Development, Validation and Application of a Curved Vortex Filament Model for Free Vortex Wake Analysis of Floating Offshore Wind Turbines," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA, Tennessee, January 2012, doi: 10.2514/6.2012-371.

[11] Bliss, D. B., Teske, M. E., and Quackenbush, T. R., "A New Methodology for Free Wake Analysis using Curved Vortex Elements," Tech. Rep. NASA-CR-3958, NASA, 1987.

[12] Van Hoydonck, W. and van Tooren, M., "Validity of Viscous Core Correction Models for Self-Induced Velocity Calculations," *arXiv.org*, April 2012, arXiv:1204.2378.

[13] Weissinger, J., "The Lift Distribution of Swept-Back Wings," Tech. Rep. 1120, NACA, March 1947.

[14] Barnes, J. P., "Semi-Empirical Vortex Step Method for the Lift and Induced Drag Loading of 2D or 3D Wings," *World Aviation Congress*, AIAA, California, October 1997, doi: 10.2514/6.1997-5559.

[15] Rosen, A. and Rand, O., "The Aerodynamics Behavior of Infinite Swept Wing: Another Point of View," *Journal of Aircraft*, Vol. 22, No. 1, January 1985, pp. 83–85, doi: 10.2514/3.45084.

[16] Wickenheiser, A. M. and Garcia, E., "Aerodynamic Modeling of Morphing Wings Using an Extended Lifting-Line Analysis," *Journal of Aircraft*, Vol. 44, No. 1, January 2007, pp. 10–16, doi: 10.2514/1.18323.

[17] Katz, J. and Plotkin, A., *Low-Speed Aerodynamics*, chap. 12, Cambridge University Press, New York, 2nd ed., 2001, pp. 331–338.

[18] McBain, G. D., *Theory of Lift*, John Wiley & Sons, Ltd., West Sussex, 2012.

[19] Phillips, W. F. and Snyder, D. O., "Modern Adaptation of Prandtl's Classic Lifting-Line Theory," *Journal of Aircraft*, Vol. 37, No. 4, July 2000, pp. 662–670, doi: 10.2514/2.2649.

[20] Weber, J. and Brebner, G., "Low-Speed Tests on 45-deg Swept-Back Wings, Part I," Tech. Rep. 2882, Ministry of Supply Aeronautical Research Council, May 1958.

[21] Celik, I. B., Ghia, U., Roache, P. J., Freitas, C. J., Coleman, H., and Raad, P. E., "Procedure for Estimation and Reporting of Uncertainty Due to Discretizationin CFD Applications," *Journal of Fluids Engineering*, Vol. 130, American Society of Mechanical Engineers, July 2008, doi: 10.1115/1.2960953.

[22] Hodson, J. D., *Numerical Analysis and Spanwise Shape Optimization for Finite Wings of Arbitrary Aspect Ratio Wings of Arbitrary Aspect Ratio*, Ph.D. thesis, Utah State University, 2019, 7574.

[23] Snyder, D. and Memory, C., "A Modified Vortex Algorithm for Low Reynolds Number, Low Aspect-Ratio Wings," *AIAA Applied Aerodynamics Conference*, AIAA, June 2005, doi: 10.2514/6.2005-4608.

[24] Biot, J. B. and Savart, F., "Note sur le magnetisme de la pile de Volta," *Annales de Chimie et de Physique*, Vol. 15, 1820, pp. 222–223.

[25] de l'Hospital, G. F. M., *Analyse des Infiniment Petits pour l'Intelligence des Lignes Courbes*, Montalant, Paris, 1716.

[26] Brown, J. W. and Churchill, R. V., *Complex Variables and Applications*, chap. 7, McGraw-Hill, New York, 8th ed., 2009.

[27] Kreysig, E., *Advanced Engineering Mathematics*, chap. 15, John Wiley & Sons, Inc., New York, 8th ed., 1999, pp. 787–793.

[28] Kuchemann, D., "A Simple Method for Calculating the Span and Chordwise Loading on Straight and Swept Wings of any Given Aspect Ratio at Subsonic Speeds," Tech. Rep. 2935, Ministry of Supply Aeronautical Research Council, August 1956.

[29] Hunsaker, D. and Phillips, W. F., "A Numerical Lifting-Line Method Using Horseshoe Vortex Sheets," *Utah Space Grant Consortium*, NASA, May 2011.

[30] Wood, D. H. and Li, D., "Assessment of the Accuracy of Representing a Helical Vortex by Straight Segments," *AIAA Journal*, Vol. 40, No. 4, April 2002, pp. 647–651, doi: 10.2514/2.1721.

[31] Gupta, S. and Leishman, J. G., "Accuracy of the Induced Velocity from Helicoidal Wake Vortices Using Straight-Line Segmentation," *AIAA Journal*, Vol. 43, No. 1, January 2005, pp. 29–40, doi: 10.2514/1.1213.

[32] Gallay, S. and Laurendeau, E., "Preliminary-Design Aerodynamic Model for Complex Configurations Using Lifting-Line Coupling Algorithm," *Journal of Aircraft*, Vol. 53, No. 4, July 2016, pp. 1145–1159, doi: 10.2514/1.C033460.

[33] Montgomery, Z. and Hunsaker, D. F., "A Propeller Model Based on a Modern Numerical Lifting-Line Algorithm with an Iterative Semi-Free Wake Solver," *AIAA SciTech Forum*, AIAA, Florida, January 2018, doi: 10.2514/6.2018-1264.

[34] Govindarajan, B. M. and Leishman, J. G., "Curvature Corrections to Improve the Accuracy of Free-Vortex Methods," *Journal of Aircraft*, Vol. 53, No. 2, March 2016, pp. 378–386, doi: 10.2514/1.C033392.

[35] Van Hoydonck, W., Gerritsma, M., and van Tooren, M., "On Core and Curvature Corrections used in Straight-Line Vortex Filament Methods," *arXiv.org*, April 2012, arXiv:1204.2699.

[36] Bhagwat, M. J. and Leishman, J. G., "Self-Induced Velocity of a Vortex Ring Using Straight-Line Segmentation," *Journal of the American Helicopter Society*, Vol. 59, No. 1, January 2014, pp. 1–7, doi: 10.4050/JAHS.59.012004.

[37] Kim, C.-J., Park, S. H., Sung, S. K., and Jung, S.-N., "Dynamic Modeling and Analysis of Vortex Filament Motion Using a Novel Curve-Fitting Method," *Chinese Journal of Aeronautics*, Vol. 29, No. 1, February 2016, pp. 53–65, doi: 10.1016/j.cja.2015.12.019.

[38] Yoon, S. S. and Heister, S. D., "Analytical Formulas for the Velocity Field Induced by an Infinitely Thin Vortex Ring," *International Journal for Numerical Methods in Fluids*, Vol. 44, No. 6, February 2004, pp. 665–672, doi: 10.1002/fld.666.

[39] Nagati, M., Iversen, J., and Vogel, J., "Vortex Sheet Modeling with Curved Higher-Order Panels," *Journal of Aircraft*, Vol. 24, No. 11, November 1987, pp. 776–782, doi: 10.2514/3.45520.

[40] Gradshteyn, I. S. and Ryzhik, I. M., *Table of Integrals, Series, and Products*, Academic Press, Oxford, 8th ed., 2014.

[41] Carlson, B. C., "Elliptic Integrals: Symmetry and Symbolic Integration," *Tricomi's Ideas and Contemporary Applied Mathematics*, Turin, December 1997.

[42] Kodaira, K., "On Compact Complex Analytic Surfaces, I," *Annals of Mathematics*, Vol. 71, No. 1, January 1960, pp. 111–152, doi: 10.2307/1969881.

[43] Kodaira, K., "On Compact Analytic Surfaces: II," *Annals of Mathematics*, Vol. 77, No. 3, May 1963, pp. 563–626, doi: 10.2307/1970131.

[44] Kodaira, K., "On Compact Analytic Surfaces: III," *Annals of Mathematics*, Vol. 78, No. 1, July 1963, pp. 1–40, doi: 10.2307/1970500.

[45] Moorthamers, B. and Hunsaker, D. F., "Accuracy of Küchemanns Prediction for the Locus of Aerodynamic Centers on Swept Wings," *AIAA SciTech Forum*, AIAA, Florida, January 2020, doi: 10.2514/6.2020-0533.

[46] Kuchemann, D. and Weber, J., "The Subsonic Flow Past Swept Wings at Zero Lift Without and With Body," Tech. Rep. 2908, Ministry of Supply Aeronautical Research Council, March 1953.

[47] Vos, R. and Farokhi, S., *Introduction to Transonic Aerodynamics*, chap. 8, Springer, Dordrecht, 2015, pp. 427–511.

[48] Kundu, P. K., Cohen, I. M., and Dowling, D. R., *Fluid Mechanics*, chap. 6 and 14, Academic Press, Massachusetts, 5th ed., 2012, pp. 197–251 and 691–728.

[49] Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," *Low Reynolds Number Aerodynamics*, edited by T. J. Mueller, Vol. 54 of *Lecture Notes in Engineering*, Springer, Berlin, 1989, pp. 1–12, doi: 10.1007/978-3-642-84010-4_1.

[50] Phillips, W. F., Fugal, S. R., and Spall, R. E., "Minimizing Induced Drag with Wing Twist, Computational-Fluid-Dynamics Validation," *Journal of Aircraft*, Vol. 43, No. 2, March 2006, pp. 437–444, doi: 10.2514/1.15089.

[51] Acton, F. S., *Numerical Methods that Work*, chap. 15, The Mathematical Association of America, USA, 1990, pp. 410–430.

[52] Magnus, A. E. and Epton, M. A., "PAN AIR: A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations using a Higher Order Panel Method. Volume 1: Theory Document (version 1.1)," Tech. Rep. CR-3251, NASA, Washington, November 1981.

[53] McCroskey, W. J., "A Critical Assessment of Wind Tunnel Results for the NACA 0012 Airfoil," Tech. Rep. TM-100019, NASA, California, October 1987.

[54] Slater, J. W., "Examining Spatial (Grid) Convergence," July 2008, https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html [retreived April 2020].

[55] Chapra, S. C. and Cananle, R. P., *Numerical Methods for Engineers*, chap. 22, McGraw-Hill, New York, 6th ed., 2010, pp. 632–636.

APPENDICES

APPENDIX A

Grid Convergence

The fundamental principle underlying every spatially-discretized numerical method is that, as the average distance between discretized nodes decreases to zero, the solution produced by the numerical method approaches the "true" value. Because the use of a large number of nodes can be computationally expensive, there is a trade-off between the accuracy of the solution and the computational cost of obtaining that solution. To this end, Richardson extrapolation is used to determine the acceptable balance between cost and accuracy [21, 54, 55].

In Richardson extrapolation, results of a numerical method are obtained using several different node spacings. Based on the change in the results of these preliminary cases, taking into account the difference in node spacing, a more-accurate solution is predicted. This practice can be demonstrated simply using a trapezoidal integration rule [55], as follows.

The true integral of a function, I, can be expressed by the sum of the approximation made using the trapezoidal rule with a step size h, $I(h)$, and the error of the estimate, $E(h)$, giving

$$I = I(h) + E(h) \tag{A.1}$$

Calculating an approximation of the integral using two step sizes, $h_1 < h_2$, the following equality is formed

$$I(h_1) + E(h_1) = I(h_2) + E(h_2) \tag{A.2}$$

In the case of the trapezoidal rule, the error is estimated to be the second-order term of the function's Taylor series, i.e.

$$E(h) \approx -\frac{b-a}{12} h^2 \bar{f}'' \tag{A.3}$$

where a and b are the bounds of the integration. Thus, assuming $\bar{f}''$ is irrespective of step size, the following ratio is found

$$E(h_1)/E(h_2) \approx h_1^2/h_2^2 \tag{A.4}$$

Combining Eq. (A.2) and Eq. (A.4), the error of the second approximation can be written as

$$E(h_1) \approx \frac{I(h_1) - I(h_2)}{(h_2/h_1)^2 - 1} \tag{A.5}$$

Thus, the integral from Eq. (A.1) is approximated as

$$I \approx I(h_1) + \frac{I(h_1) - I(h_2)}{(h_2/h_1)^2 - 1} \tag{A.6}$$

providing a higher-order estimate of the desired integral.

In this example, the trapezoidal rule is known to be second-order convergent. If the order of convergence of the method is not known, a similar technique can be used, though, results obtained using three node spacings are needed instead of two. The three sets of results are used to calculate the apparent order of convergence, p, which is then used to extrapolate the results, approximating the solution at an infinitesimally small average node spacing [21, 54]. In this case, $E(h)$ is assumed to be of the form

$$E(h) \approx Ch^p \tag{A.7}$$

where C is a constant, h is the average distance between computational nodes, and p is the order of convergence. This model of the error leads to a formula similar to Eq. (A.6)

$$I \approx I(h_1) + \frac{I(h_1) - I(h_2)}{(h_2/h_1)^p - 1} \tag{A.8}$$

where p is found by taking the natural logarithm of Eq. (A.7)

$$p \approx \frac{\ln(E(h)/C)}{\ln(h)} \tag{A.9}$$

In Eq. (A.9) it is observed that p is the slope of a logarithmic line of the error as a function of the average step size. Therefore, Eq. (A.9) may be approximated using the error estimates obtained with the step sizes $h_a < h_b$

$$p \approx \frac{\ln(E(h_b)/C) - \ln(E(h_a)/C)}{\ln(h_b) - \ln(h_a)} \tag{A.10}$$

With the final term on the right hand side of Eq. (A.8) as an estimate of $E(h_a)$ and $E(h_b)$, and rearranging Eq. (A.10), p can be approximated using the results obtained using step sizes $h_1 < h_2 < h_3$ as

$$p \approx \frac{\ln\left(\frac{I(h_3)-I(h_2)}{I(h_2)-I(h_1)}\right) + \ln\left(\frac{(h_2/h_1)^p-1}{(h_3/h_2)^p-1}\right)}{\ln(h_2/h_1)} \tag{A.11}$$

For the special case of $h_2/h_1 = h_3/h_2$, Eq. (A.11) simplifies to

$$p \approx \frac{\ln\left(\frac{I(h_3)-I(h_2)}{I(h_2)-I(h_1)}\right)}{\ln(h_2/h_1)} \tag{A.12}$$

This estimate for p is then used in Eq. (A.8) to calculate the extrapolated value.

APPENDIX B

Source Code

B.1 Vortex Panel Method

```
import numpy as np
from scipy.linalg import lu_factor, lu_solve


class VPM():
    '''Calculates the circulation distribution on an airfoil's surface.

    From a series of x and y locations, describing the vertices of an arbitrary
    number of vortex panels with linearly varying strength, the strengths of
    each panel is calculated such that the normal velocity at each panel's
    control point is zero and the Kutta Condition is satisfied at the trailing
    edge.

    *Methods in this class allow the infinite wing (airfoil) to have a sweep
    angle.

    Attributes
    ----------
    control_xy : 2D array
        Array of size [2,n-1], containing the x and y coordinates of vortex
        panel control points, where n is the number of vertices
        (self.node_count).

    node_count : scalar
        The number of vortex panel vertices.

    sweep : scalar
        Sweep angle of the infinite wing, in radians. Should be scaled by
        "-1" if the analyzed section is located on a left-hand wing. This
        convention is used to determine the proper effect of the sideslip
        angle.

    vortex_xy : 2D array
        Array of size [2,n], containing the x and y coordinates of vortex panel
        vertexes, where n is the number of vertexes (self.node_count).

    Methods
    -------
    reinitialize :
        Resets the vortex panel class attributes from parameters.

    solve :
        Solves the vortex strengths, and lift and moment coefficients.
```

```
    Notes
    -----
    This class assumes a clockwise order to the vertex coordinates, beginning
    at the trailing edge.

    All inputs should be given as unswept values. All necessary
    modifications due to sweep are accounted for internally.

    The aerodynamic coordinate system is used in this method.
        (i.e. x-axis points from LE to TE of an unswept airfoil,
              y-axis points "up" in the plane of an unswept airfoil,
          and z-axis points in the direction cross(x,y))

    Examples
    --------
    This class may be used as follows:

    >>> my_VPM = VPM(my_x_coordinates, my_y_coordinates)
    >>> my_results = my_VPM.solve(my_flow_vector)

    '''

    def __init__(self, x, y, sweep=0., control_xy=None):
        '''Sets vortex panel class attributes from parameters.

        Parameters
        ----------
        x : array_like
            x coordinates of vortex panel vertexes (un-swept), len(x) = n.

        y : array_like
            y coordinates of vortex panel vertexes, len(y) = n.

        sweep : scalar, optional
            Sweep angle of the infinite wing, in radians. Should be scaled by
            "-1" if the analyzed section is located on a left-hand wing. This
            convention is used to determine the proper effect of the sideslip
            angle.

        control_xy : 2D array, optional
            Array of size [2,n-1], containing the x and y coordinates of vortex
            panel control points.
        '''

        self.reinitialize(x, y, sweep, control_xy)

    def reinitialize(self, x, y, sweep=0., control_xy=None):
        '''Sets vortex panel class attributes from parameters.

        Parameters
        ----------
        x : array_like
            x coordinates of vortex panel vertexes (un-swept), len(x) = n.
```

```
        y : array_like
            y coordinates of vortex panel vertexes, len(y) = n.

        sweep : scalar, optional
            Sweep angle of the infinite wing, in radians. Should be scaled by
            "-1" if the analyzed section is located on a left-hand wing. This
            convention is used to determine the proper effect of the sideslip
            angle.

        control_xy : 2D array, optional
            Array of size [2,n-1], containing the x and y coordinates of vortex
            panel control points.
        '''

        self.sweep = sweep
        self.node_count = len(x)
        self.vortex_xy = np.array([x*np.cos(sweep), y])
        if control_xy is not None:
            self.control_xy = np.array([control_xy[0]*np.cos(sweep),
                                        control_xy[1]])
        else:
            self.control_xy = (self.vortex_xy[:, 1:]
                               + self.vortex_xy[:, :-1])/2.

        self._panel_length = np.sqrt((self.vortex_xy[0, 1:]
                                      - self.vortex_xy[0, :-1])**2
                                     + (self.vortex_xy[1, 1:]
                                      - self.vortex_xy[1, :-1])**2)

        _P11, _P12, _P21, _P22 = self._p_matrix(*self.control_xy,
                                                *self.vortex_xy[:, :-1],
                                                *self.vortex_xy[:, 1:],
                                                self._panel_length)

        self._A = np.zeros((self.node_count, self.node_count))

        self._A[:-1, :-1] += (self.vortex_xy[0, 1:, None]
                              - self.vortex_xy[0, :-1, None]) \
            * _P21/self._panel_length[:, None] \
            - (self.vortex_xy[1, 1:, None]
               - self.vortex_xy[1, :-1, None])*_P11/self._panel_length[:, None]

        self._A[:-1, 1:] += (self.vortex_xy[0, 1:, None]
                             - self.vortex_xy[0, :-1, None]) \
            * _P22/self._panel_length[:, None] \
            - (self.vortex_xy[1, 1:, None]
               - self.vortex_xy[1, :-1, None])*_P12/self._panel_length[:, None]

        self._A[-1, 0] = 1.
        self._A[-1, -1] = 1.

        self._lu_decomposition = lu_factor(self._A, overwrite_a=True)

    def solve(self, V, chord=1., Vref=1.):
        '''Solves the vortex strengths, and lift and moment coefficients.
```

```
    Parameters
    ----------
    V : array_like
        An array of size [3] containing the x, y, and z components of the
        free-stream velocity.

    chord : scalar, optional
        The chord length of the un-swept airfoil, used for
        non-dimensionalization.

    Vref : scalar, optional
        The magnitude of the un-swept freestream, used for
        non-dimensionalization.

    Returns
    -------
    C_L : scalar
        Lift coefficient of the airfoil.

    C_m : scalar
        Moment coefficient of the airfoil about the leading edge of the
        swept infinite wing.

    total_circulation : scalar
        The integrated total circulation of the airfoil.

    '''

    alpha, Vinf = self._sweep_vector(self.sweep, V)

    self._b = np.zeros(self.node_count)

    self._b[:-1] = (Vinf/self._panel_length) \
        * ((self.vortex_xy[1, 1:] - self.vortex_xy[1, :-1])*np.cos(alpha)
           - (self.vortex_xy[0, 1:]
              - self.vortex_xy[0, :-1])*np.sin(alpha))

    self._vortex_strengths = lu_solve(self._lu_decomposition, self._b)

    total_circulation = np.sum(self._panel_length
                               * (self._vortex_strengths[:-1]
                                  + self._vortex_strengths[1:])/2.)

    C_L = 2.*Vinf*total_circulation/(chord*np.cos(self.sweep)*Vref**2)

    C_P = 1. - (self._vortex_strengths/Vref)**2

    C_m = -np.sum(((2.*self.vortex_xy[0, :-1]*self._vortex_strengths[:-1]
                    + self.vortex_xy[0, :-1]*self._vortex_strengths[1:]
                    + self.vortex_xy[0, 1:]*self._vortex_strengths[:-1]
                    + 2.*self.vortex_xy[0, 1:]*self._vortex_strengths[1:])
                   * np.cos(alpha)
                   + (2.*self.vortex_xy[1, :-1]*self._vortex_strengths[:-1]
                      + self.vortex_xy[1, :-1]*self._vortex_strengths[1:]
```

```
                    + self.vortex_xy[1, 1:]*self._vortex_strengths[:-1]
                    + 2.*self.vortex_xy[1, 1:]*self._vortex_strengths[1:])
                * np.sin(alpha))*self._panel_length)\
        * Vinf/(3.*chord*chord*np.cos(self.sweep)*Vref**2)

    return C_L, C_m, total_circulation

def _angles_from_vector(self, V):
    '''Defines flow angles and magnitude from a flow vector.

    Parameters
    ----------
    V : array_like
        An array of size [3] containing the x, y, and z components of the
        free-stream velocity.

    Returns
    -------
    alpha : scalar
        Angle between the free-stream and the x-z plane (radians).

    beta : scalar
        Angle between the free-stream and the x-y plane (radians).

    Vinf : scalar
        The magnitude of the free-stream velocity.
    '''

    _V0, _V1, _V2 = V

    # alpha, beta, Vinf
    return np.arctan2(_V1, _V0), np.arctan2(_V2, _V0), \
        np.sqrt(_V0*_V0 + _V1*_V1 + _V2*_V2)

def _p_matrix(self, xc, yc, x, y, x1, y1, l):
    '''Calculates the panel coefficient matrix ("P Matrix").

    Parameters
    ----------
    xc, yc : coordinates at which the panel's influence is desired

    x, y : coordinates of the starting vertex of the panel

    x1, y1 : coordinates of the ending vertex of the panel

    l : length of the panel

    Returns
    -------
    P : The elements of the [2,2] P matrix, returned as four scalars.
    '''

    _xi = (1./l)*((x1[None, :]-x[None, :])*(xc[:, None]-x[None, :])
                  + (y1[None, :]-y[None, :])*(yc[:, None]-y[None, :]))
    _eta = (1./l)*(-(y1[None, :]-y[None, :])*(xc[:, None]-x[None, :])
```

```
                    + (x1[None, :]-x[None, :])*(yc[:, None]-y[None, :]))

        _Phi = np.arctan2((_eta*l), (_eta*_eta + _xi*_xi - _xi*l))
        _Psi = .5*np.log((_eta*_eta + _xi*_xi)/(_eta*_eta
                                                + (_xi - l)*(_xi - l)))

        _constant = (2.*np.pi*l*l)
        _XY11 = (x1[None, :]-x[None, :])/_constant
        _XY12 = -(y1[None, :]-y[None, :])/_constant
        _XY21 = (y1[None, :]-y[None, :])/_constant
        _XY22 = (x1[None, :]-x[None, :])/_constant

        _P11 = (l - _xi)*_Phi + _eta*_Psi
        _P12 = _xi*_Phi - _eta*_Psi
        _P21 = _eta*_Phi - (l-_xi)*_Psi - l
        _P22 = -_eta*_Phi - _xi*_Psi + l

        return _XY11*_P11 + _XY12*_P21, _XY11*_P12 + _XY12*_P22, \
            _XY21*_P11 + _XY22*_P21, _XY21*_P12 + _XY22*_P22

    def _sweep_vector(self, angle, V):
        '''Effective 2D flow properties for a sweep angle from flow vector.

        Parameters
        ----------
        V : array_like
            An array of size [3] containing the x, y, and z components of the
            free-stream velocity.

        angle : scalar
            The sweep angle at which the effective airfoil coordinates will be
            calculated (in radians).

        Returns
        -------
        alpha : scalar
            Effective angle of attack (in radians).

        Vinf : scalar
            Effective free-stream velocity.

        '''
        _alpha, _beta, _Vinf = self._angles_from_vector(V)

        _Ca = np.cos(_alpha)
        _Sa = np.sin(_alpha)
        _Cb = np.cos(_beta)
        _Sb = np.sin(_beta)
        _sqrtSaSb = np.sqrt(1. - _Sa*_Sa*_Sb*_Sb)

        alpha = np.arctan2(np.tan(_alpha)*_Cb,
                           np.cos(angle - _beta))

        Vinf = _Vinf*np.sqrt(_Ca*_Ca*np.cos(angle - _beta)**2
                             + _Sa*_Sa*_Cb*_Cb)/_sqrtSaSb
```

```
        return alpha, Vinf
```

B.2 General Implementation of Lifting-Line Theory

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import fsolve
from Vortex_Panel_Method import VPM


class LL():
    '''Calculates the circulation distribution across a finite, planar wing.

    This method predicts the circulation and force distributions of wings with
    uniform sweep and taper (+ elliptic), and a NACA 4-digit cross-section.


    Attributes
    ----------
    area : scalar
        Planform area of the wing.

    airfoil_aL0 : scalar
        The zero-lift angle of attack of the root airfoil (rad).

    airfoil_CLa : scalar
        The lift slope of the root airfoil (1/rad).

    airfoil_ctrl_xy : 2D array
        Array of size [2,airfoil_n-1], containing the x and y coordinates
        of vortex panel control points, used in the vortex panel method.

    airfoil_n : int
        Total number of vertices around the airfoil.

    airfoil_name : str
        String containing the NACA 4-digit airfoil designation ('XXXX').

    airfoil_xy : 2D array
        An array of size [2,airfoil_n] that contains the x and y coordinates
        of locations on the unswept airfoil's surface. The points are ordered
        such that they begin at the trailing edge and continue clockwise around
        the airfoil, ending at the trailing edge.

    blend : scalar
        The normalized blending distance, used to calculate the
        effective loci of aerodynamic centers.

    circulation_dist : array
        An array of size [wing_n] that contains the circulation distribution of
        the wing.
```

```
delta : scalar
    The fraction of the local chord the finite segment of the TV
    extends from the LAC.

local_chord : array_like
    An array of size [wing_n+1] that contains the local un-swept chord at
    each coordinate location in wing_xyz.

local_chord_ctrl : array_like
    An array of size [wing_n] that contains the local un-swept chord at
    each coordinate location in wing_ctrl_xyz.

local_dchord : array_like
    An array of size [wing_n+1] that contains the change in local un-swept
    chord at each coordinate location in wing_xyz.

local_dchord_ctrl : array_like
    An array of size [wing_n] that contains the change in local un-swept
    chord at each coordinate location in wing_ctrl_xyz.

local_sweep : array_like
    An array of size [wing_n+1] that contains the local sweep at each
    coordinate location in wing_xyz. Negative sweep corresponds to "left-
    hand wings". This convention is used to determine the proper effect of
    the sideslip angle.

local_sweep_ctrl : array_like
    An array of size [wing_n] that contains the  local sweep at each
    coordinate location in wing_ctrl_xyz. Negative sweep corresponds to
    "left-hand wings". This convention is used to determine the proper
    effect of the sideslip angle.

local_sweep_eff : 2D array
    An array of size [wing_n,wing_n+1] that contains the effective local
    sweep at each coordinate location in wing_xyz_eff. Negative sweep
    corresponds to "left-hand wings". This convention is used to determine
    the proper effect of the sideslip angle.

RA : scalar
    The aspect ratio to the wing.

root_chord : scalar
    Length of the root airfoil's chord.

section_area : array_like
    An array of size [wing_n] that contains the area of each wing section.

section_length : array_like
    An array of size [wing_n] that contains the length of each bound
    vortex.

section_vector : array_like
    An array of size [3,wing_n] that contains the vector of each bound
    vortex.
```

```
span : scalar
    Distance from wing-tip to wing-tip along the z-axis.

sweep : scalar
    The wing's angle of sweep (rad).

taper : scalar
    Ratio of the tip airfoil's chord to the root airfoil's chord.
    (taper = -1 results in an elliptic wing)

u : array_like
    An array of size [3] containing the x, y, and z components of the
    normalized free-stream velocity, V.

V : array_like
    An array of size [3] containing the x, y, and z components of the
    free-stream velocity.

Vinf : scalar
    The magnitude of the free-stream velocity, V.

wing_ctrl_xyz : 2D array
    An array of size [3,wing_n] that contains the x, y, and z coordinates
    of the control point locations on the LAC.

wing_joint_xyz :
    An array of size [3,wing_n+1] that contains the x, y, and z coordinates
    of the joint locations formed by the vortex segments and semi-infinite
    vortices that originate from the LAC.

wing_joint_xyz_eff :
    An array of size [3,wing_n,wing_n+1] that contains the x, y, and z
    coordinates of the effective joint locations formed by the vortex
    segments and semi-infinite vortices that originate from the effective
    LACs.

wing_n : int
    Total number of control nodes along the wing.

wing_xyz : 2D array
    An array of size [3,wing_n+1] that contains the x, y, and z coordinates
    of endpoint locations for the bound vortex segments along the LAC

wing_xyz_eff : 2D array
    An array of size [3,wing_n,wing_n+1] that contains the x coordinates of
    endpoint locations on the bound vortex segments along the effective
    LACs.

Methods
-------
plot_wing :
    A plot of the top view of a planar wing planform.

solve :
    Executes the lifting-line algorithm to predict the circulation
```

```
    distribution along the wing.

Notes
-----
The aerodynamic coordinate system is used in this method.
    (i.e. x-axis points from LE to TE of the root airfoil,
          y-axis points "up" in the plane of the root airfoil,
      and z-axis points in the direction cross(x,y))
The coordinate system's origin is located at the leading edge of the root
airfoil.

Examples
--------
This class may be used as follows:

>>> my_LL = Lifting_Line.LL()
>>> my_LL.plot_wing(Title='My Wing')
>>> my_results = my_LL.solve()

'''

def __init__(self, airfoil_n=199, airfoil_name='2412', airfoil_cluster=1,
             airfoil_ctrl_cluster=0, openTE=False, wing_n=99, span=8.,
             root_chord=1., taper=1., sweep=0., wing_cluster=2,
             wing_ctrl_cluster=1, blend=0.25, delta=0.15, V=[1., 0., 0.],
             half=False, aero_approx=True, aero_props=None, LAC_mod='k',
             v_core=0.0, thin_approx=False):
    '''Initialization and set-up of the lifting-line model.

    Parameters
    ----------
    airfoil_n : int, optional
        Total number of control points around the airfoil.

    airfoil_name : str, optional
        String containing the NACA 4-digit airfoil designation ('XXXX').

    airfoil_cluster : int, optional
        Defines the node distribution (0 = even distribution,
                                       1 = cluster at LE and TE).

    airfoil_ctrl_cluster : int, optional
        Defines the control point distribution (0 = mid-point of panel,
                                                1 = cluster at LE and TE).

    openTE : bool, optional
        Flag indicating if the airfoil trailing edge should be closed
        (False) or open (True).

    wing_n : int, optional
        Total number of control nodes along the wing.

    span : scalar, optional
        Distance from wing-tip to wing-tip along the z-axis.
```

```
root_chord : scalar, optional
    Length of the root airfoil's chord.

taper : scalar, optional
    Ratio of the tip airfoil's chord to the root airfoil's chord.
    (taper = -1 results in an elliptic wing)

sweep : scalar, optional
    The wing's angle of sweep (rad).

wing_cluster : int, optional
    Defines the node distribution (0 = even distribution,
                                   1 = cluster at tips,
                                   2 = cluster at tips and root).

wing_ctrl_cluster : int, optional
    Defines the control point distribution (0 = mid-point of panel,
                                            1 = same as wing_cluster).

blend : scalar, optional
    The normalized blending distance, used to calculate the
    effective loci of aerodynamic centers.

delta : scalar, optional
    The fraction of the local chord the vortex segment portion of the
    TV extends from the LAC.

V : array_like, optional
    An array of size [3] containing the x, y, and z components of the
    free-stream velocity used for initialization.

half : bool, optional
    Defines if the wing is one half wing (True) or a full wing (False).

aero_approx : bool, optional
    Defines if the curve-fit airfoil should be used to approximate
    section propeties (True), or if the vortex panel method should
    be used directly (False).

aero_props : array_like, optional
    An array of size [2] containing the desired values for airfoil_CLa
    (1/rad) and airfoil_aL0 (rad). These values will be used for every
    wing section. If "None", CLa and aL0 are calculated according to
    aero_approx.

LAC_mod : str, optional
    Flag describing the model to be used for the locus of aerodynamic
    centers ('k' = Kuchemann, 'c' = quarter-Chord, 'w' = Weissinger).
    Weissinger's method, instead of lifting-line theory, is applied
    if LAC_mod = 'w'.

v_core : scalar, optional
    The radius of the vortex finite core.

thin_approx : bool, optional
```

```
    Defines if the thin-airfoil approximation for swept-wing section
    properties is used (True), or if the generalized section properties
    are used (False).
'''

# Set flag for Weissenger method
if LAC_mod == 'w':
    self._weiss = True
else:
    self._weiss = False

# Calculate airfoil nodes
self._NACA_4(airfoil_n, airfoil_name, airfoil_cluster,
             airfoil_ctrl_cluster, openTE)

# Set airfoil aerodynamic information flags
if aero_props is not None:
    self._aero_approx = True
    self.airfoil_CLa = aero_props[0]
    self.airfoil_aL0 = aero_props[1]
else:
    self._aero_approx = aero_approx
    self.airfoil_CLa, self.airfoil_aL0 = self._aero_properties_calc()

# Set wing properties
self.wing_n = wing_n
self.span = span
self.root_chord = root_chord
self.taper = taper
self.sweep = sweep
self.blend = blend
sigma = 4.*np.cos(sweep)**2/(blend*blend*span*span)
self.delta = delta
self.V = V
self.Vinf = np.sqrt(V[0]*V[0] + V[1]*V[1] + V[2]*V[2])
self.u = self._normalize(self.V)
self.v_core = v_core
self.thin_approx = thin_approx
if taper == -1.:
    self.area = root_chord*span*np.pi/4.
else:
    self.area = root_chord*span*(1 + taper)/2.
self.RA = span*span/self.area

# Calculate z-coordinates
self.wing_xyz = np.zeros((3, wing_n + 1))
self.wing_ctrl_xyz = np.zeros((3, wing_n))

if wing_cluster == 0:
    if half:
        _theta_bound = np.linspace(0, span/2., 2*wing_n + 1)
    else:
        _theta_bound = np.linspace(-span/2., span/2., 2*wing_n + 1)

    self.wing_xyz[2, :] = _theta_bound[::2]
```

```
        self.wing_ctrl_xyz[2, :] = _theta_bound[1::2]

    elif wing_cluster == 1:
        if half:
            _theta_bound = np.linspace(0, np.pi, 2*wing_n + 1)
        else:
            _theta_bound = np.linspace(-np.pi/2, np.pi/2, 2*wing_n + 1)

        self.wing_xyz[2, :] = 0.5*self.span*np.sin(_theta_bound[::2])
        self.wing_ctrl_xyz[2, :] = 0.5*self.span*np.sin(_theta_bound[1::2])

    elif wing_cluster == 2:
        if half:
            _theta_bound = np.linspace(np.pi, 2*np.pi, 2*wing_n + 1)
        else:
            _theta_bound = np.linspace(0, 2*np.pi, 2*wing_n + 1)

        self.wing_xyz[2, :] = np.sign(_theta_bound[::2] - np.pi) \
            * 0.25*self.span*(1 + np.cos(_theta_bound[::2]))
        self.wing_ctrl_xyz[2, :] = np.sign(_theta_bound[1::2] - np.pi) \
            * 0.25*self.span*(1 + np.cos(_theta_bound[1::2]))

    if wing_ctrl_cluster == 0:
        self.wing_ctrl_xyz[2, :] = (self.wing_xyz[2, 1:]
                                    + self.wing_xyz[2, :-1])/2.

    # Calculate chord distribution
    self.local_chord = np.zeros(wing_n + 1)
    self.local_chord_ctrl = np.zeros(wing_n)
    self.local_dchord = np.zeros(wing_n + 1)
    self.local_dchord_ctrl = np.zeros(wing_n)

    if self.taper == -1.:
        self.local_chord = \
            root_chord*np.sqrt(1. - (2.*0.999999*self.wing_xyz[2, :]
                               / span)**2)
        self.local_chord_ctrl = \
            root_chord*np.sqrt(1. - (2.*0.999999*self.wing_ctrl_xyz[2, :]
                               / span)**2)

        self.local_dchord = -4.*root_chord*0.999999*self.wing_xyz[2, :] \
            / (np.sqrt(1. - (2.*0.999999*self.wing_xyz[2, :]/span)**2)
               * span*span)
        self.local_dchord_ctrl = \
            -4.*root_chord*0.999999*self.wing_ctrl_xyz[2, :] \
            / (np.sqrt(1. - (2.*0.999999*self.wing_ctrl_xyz[2, :]/span)**2)
               * span*span)
    else:
        self.local_chord = \
            root_chord*(1. - 2.*(1. - taper)
                        * abs(self.wing_xyz[2, :])/span)
        self.local_chord_ctrl = \
            root_chord*(1. - 2.*(1. - taper)
                        * abs(self.wing_ctrl_xyz[2, :])/span)
```

```
        self.local_dchord = \
            2.*root_chord*(taper - 1.) \
            * np.sign(self.wing_xyz[2, :])/span
        self.local_dchord_ctrl = \
            2.*root_chord*(taper - 1.) \
            * np.sign(self.wing_ctrl_xyz[2, :])/span

    # Calculate x-coordinates
    if LAC_mod == 'k':
        self.wing_xyz[0, :] = self._kuchemann(self.wing_xyz[2, :],
                                              self.local_chord, root_chord,
                                              sweep, span)

        self.wing_ctrl_xyz[0, :] = \
            self._kuchemann(self.wing_ctrl_xyz[2, :],
                            self.local_chord_ctrl, root_chord, sweep, span)

    elif LAC_mod == 'c':
        self.wing_xyz[0, :] = self._quarter_chord(self.wing_xyz[2, :],
                                                  root_chord, sweep)

        self.wing_ctrl_xyz[0, :] = \
            self._quarter_chord(self.wing_ctrl_xyz[2, :], root_chord,
                                sweep)

    elif LAC_mod == 'w':
        self.wing_xyz[0, :] = self._quarter_chord(self.wing_xyz[2, :],
                                                  root_chord, sweep)

        self.wing_ctrl_xyz[0, :] = \
            self._quarter_chord(self.wing_ctrl_xyz[2, :], root_chord,
                                sweep) + 0.5*self.local_chord_ctrl

    self.wing_xyz_eff = np.zeros((3, wing_n, wing_n + 1))
    self.wing_xyz_eff[0, :, :] = \
        self._f_cond_cav(self.wing_xyz[2, None, :],
                         self.wing_ctrl_xyz[2, :, None],
                         self.local_chord[None, :],
                         self.local_chord_ctrl[:, None],
                         self.local_dchord[None, :],
                         self.local_dchord_ctrl[:, None],
                         sigma, sweep, root_chord, span, LAC_mod)
    self.wing_xyz_eff[2, :, :] = self.wing_xyz[2, None, :]

    # Calculate local sweep
    if LAC_mod == 'k':
        self.local_sweep = -np.arctan(self._dkuchemann(self.wing_xyz[2, :],
                                                       self.local_chord,
                                                       self.local_dchord,
                                                       sweep, span))

        self.local_sweep_ctrl = \
            -np.arctan(self._dkuchemann(self.wing_ctrl_xyz[2, :],
                                        self.local_chord_ctrl,
                                        self.local_dchord_ctrl,
```

```
                                    sweep, span))

elif LAC_mod == 'c' or LAC_mod == 'w':
    self.local_sweep = \
        -np.arctan(self._dquarter_chord(self.wing_xyz[2, :], sweep))

    self.local_sweep_ctrl = \
        -np.arctan(self._dquarter_chord(self.wing_ctrl_xyz[2, :],
                                        sweep))

self.local_sweep_eff = \
    -np.arctan(self._df_cond_cav(self.wing_xyz[2, None, :],
                                 self.wing_ctrl_xyz[2, :, None],
                                 self.local_chord[None, :],
                                 self.local_chord_ctrl[:, None],
                                 self.local_dchord[None, :],
                                 self.local_dchord_ctrl[:, None],
                                 sigma, sweep, root_chord, span,
                                 LAC_mod))

# Calculate other section properties
self.section_vector = self.wing_xyz[:, :-1] - self.wing_xyz[:, 1:]

self.section_length = np.linalg.norm(self.section_vector, axis=0)

self.section_area = 0.5*(self.local_chord[:-1]
                         + self.local_chord[1:]) \
    * abs(self.wing_xyz[2, :-1] - self.wing_xyz[2, 1:])

self._zeta = self.section_vector/self.section_area

if self._aero_approx:
    if thin_approx:
        self._aero_properties = \
            np.array([self.airfoil_CLa*np.cos(self.local_sweep_ctrl),
                      self.airfoil_aL0 + 0.*self.local_sweep_ctrl])

    elif aero_props is not None:
        _thick = float(airfoil_name[2:])/100.
        _R_CLa = 1./(1. - 0.2955*_thick**0.96*self.local_sweep_ctrl**2
                     - 0.1335*_thick**0.68*self.local_sweep_ctrl**4)
        _camber = float(airfoil_name[0])/100.
        _d_aL0 = 1./(1. + 0.5824*_camber**0.92*self.local_sweep_ctrl**2
                     + 1.3892*_camber**1.16*self.local_sweep_ctrl**4) \
            - 1.
        self._aero_properties = \
            np.array([_R_CLa*aero_props[0], aero_props[1] + _d_aL0])
    else:
        self._aero_properties = \
            np.array(np.transpose([self._aero_properties_calc(sweep)
                                   for sweep in
                                   self.local_sweep_ctrl]))
else:
    self._vortex_panel_method_objects = \
        [VPM(*self.airfoil_xy, sweep, self.airfoil_ctrl_xy)
```

```
            for sweep in self.local_sweep_ctrl]

        # Calculate TV joint locations
        if self._weiss:
            self.wing_joint_xyz = np.zeros((3, wing_n + 1))
            self.wing_joint_xyz_eff = np.zeros((3, wing_n, wing_n + 1))

            self.wing_joint_xyz[0, :] = self.wing_xyz[0, :] \
                + delta*self.local_chord

            self.wing_joint_xyz_eff[0, :, :] = self.wing_xyz_eff[0, :, :] \
                + delta*self.local_chord

            self.wing_joint_xyz[2, :] = self.wing_xyz[2, :]

            self.wing_joint_xyz_eff[2, :, :] = self.wing_xyz[2, None, :]

        else:
            self.wing_joint_xyz = np.zeros((3, wing_n + 1))
            self.wing_joint_xyz_eff = np.zeros((3, wing_n, wing_n + 1))

            self.wing_joint_xyz[0, :] = self.wing_xyz[0, :] \
                + delta*self.local_chord*np.cos(self.local_sweep)

            self.wing_joint_xyz_eff[0, :, :] = self.wing_xyz_eff[0, :, :] \
                + delta*self.local_chord*np.cos(self.local_sweep_eff)

            self.wing_joint_xyz[2, :] = self.wing_xyz[2, :] \
                + delta*self.local_chord*np.sin(self.local_sweep)

            self.wing_joint_xyz_eff[2, :, :] = self.wing_xyz[2, None, :] \
                + delta*self.local_chord*np.sin(self.local_sweep_eff)

        return

    def plot_wing(self, title=None, TV=True, ctrl=False, eff=False, eff_n=5):
        '''A plot of the top view of a planar wing planform.

        This plot also includes the locus of aerodynamic centers, bound vortex
        vertices, control points, the finite segment of trailing vortecies,
        a finite portion of the semi-infinite vortices, and, a specified number
        of effective LACs.

        Parameters
        ----------
        title : str, optional
            A string containing the title of the figure.

        TV : bool, optional
            Flag indicating whether TVs should (True) or should not (False) be
            plotted.

        ctrl : bool, optional
            Flag indicating whether control points should (True) or should
            not (False) be plotted.
```

```
    eff : bool, optional
        Flag indicating whether effective LAC should (True) or should
        not (False) be plotted.

    eff_n : int, optional
        The step size of the iterator for the effective LAC array. This
        method will only plot every eff_nth LAC (assuming eff==True).

    Returns
    -------
    fig : obj
        matplotlib figure object.
    '''

    fig = plt.figure()

    # TVs
    if TV:
        _u = 1.5*self.root_chord*self._normalize(self.V)
        plt.plot([self.wing_xyz[2, :], self.wing_joint_xyz[2, :],
                  self.wing_joint_xyz[2, :] + _u[2]],
                 [self.wing_xyz[0, :], self.wing_joint_xyz[0, :],
                  self.wing_joint_xyz[0, :] + _u[0]], 'grey')

    # Wing planform and quarter=chord line
    plt.plot(self.wing_xyz[2, :],
             abs(self.wing_xyz[2, :])*np.tan(self.sweep)
             + 0.25*self.root_chord + 0.75*self.local_chord, 'k-')
    plt.plot(self.wing_xyz[2, :],
             abs(self.wing_xyz[2, :])*np.tan(self.sweep)
             + 0.25*self.root_chord - 0.25*self.local_chord, 'k-')
    plt.plot(self.wing_xyz[2, :],
             abs(self.wing_xyz[2, :])*np.tan(self.sweep)
             + 0.25*self.root_chord, ':', color='darkgrey')

    # Effective LACs
    if eff:
        plt.plot(np.transpose(self.wing_xyz_eff[2, ::eff_n, :]),
                 np.transpose(self.wing_xyz_eff[0, ::eff_n, :]), '--')

    # LAC
    plt.plot(self.wing_xyz[2, :], self.wing_xyz[0, :], 'k-')

    # Control points
    if ctrl:
        plt.plot(self.wing_ctrl_xyz[2, :], self.wing_ctrl_xyz[0, :], 'r+')

    plt.gca().invert_yaxis()
    plt.gca().invert_xaxis()

    plt.xlabel('z')
    plt.ylabel('x')

    if title is not None:
```

```
            plt.title(title)

        plt.axis('equal')
        plt.grid(True)

        return fig

    def solve(self, V=None, G0=None, jacobian=True):
        '''Executes the lifting-line algorithm.

        Parameters
        ----------
        V : array_like, optional
            An array of size [3] containing the x, y, and z components of the
            free-stream velocity.

        G0 : array_like, optional
            An array of size [wing_n], containing an initial guess for the
            circulation distribution.

        jacobian : bool
            Flag determining the use of the Jacobian in the non-linear solver
            (only available if _aero_approx is initialized as True).

        Returns
        -------
        CL : scalar
            The total lift coefficient. Defined as the component of total force
            in the symmetry plane that is perpendicular the freestream.

        CDi : scalar
            The total induced drag coefficient. Defined as the component of
            total force aligned with the freestream.

        CS : scalar
            The total side-force coefficient. Defined as the component of total
            force normal to CL and CDi.
        '''
        if V is not None:
            self.V = V
            self.Vinf = np.sqrt(V[0]*V[0] + V[1]*V[1] + V[2]*V[2])
            self.u = self._normalize(V)

        # Calculate the influence matrix
        _uinf = np.broadcast_to(self.u[:, None, None],
                                (3, self.wing_n, self.wing_n))

        _p1 = self.wing_joint_xyz_eff[:, :, 1:]
        _p2 = self.wing_xyz_eff[:, :, 1:]
        _p3 = self.wing_xyz_eff[:, :, :-1]
        _p4 = self.wing_joint_xyz_eff[:, :, :-1]
        _control = np.broadcast_to(self.wing_ctrl_xyz[:, :, None],
                                   (3, self.wing_n, self.wing_n))

        _bound_mask = np.ones((self.wing_n, self.wing_n)) \
```

```
        - np.diag(np.ones(self.wing_n), 0)

    core = self.v_core

    if self._weiss:
        self._TV_influence = \
            - self._straight_semi_infinite(_p1, _uinf, _control, core) \
            + self._straight_segment(_p1, _p2, _control, core) \
            + self._straight_segment(_p2, _p3, _control, core) \
            + self._straight_segment(_p3, _p4, _control, core) \
            + self._straight_semi_infinite(_p4, _uinf, _control, core)

        _A = np.dot(self._TV_influence.T, self._cam_norm).T

        _b = np.full(self.wing_n, -np.dot(self.u, self._cam_norm))

        _G = np.linalg.solve(_A, _b)

        self._TV_influence = \
            - self._straight_semi_infinite(_p1, _uinf, _control, core) \
            + self._straight_segment(_p1, _p2, _control, core) \
            + self._straight_segment(_p3, _p4, _control, core) \
            + self._straight_semi_infinite(_p4, _uinf, _control, core)

        _F = 2.*np.sum(np.cross(self._TV_influence @ _G
                                + self.u[:, None], self._zeta, axis=0)
                       * _G*self.section_area/self.area, axis=1)
        self.circulation_dist = _G*self.Vinf

    else:
        self._TV_influence = \
            - self._straight_semi_infinite(_p1, _uinf, _control, core) \
            + self._straight_segment(_p1, _p2, _control, core) \
            + self._straight_segment(_p2, _p3, _control, core) \
            * _bound_mask[None, ...] \
            + self._straight_segment(_p3, _p4, _control, core) \
            + self._straight_semi_infinite(_p4, _uinf, _control, core)

        # Solve non-linear system
        _guess = 0.5*self.root_chord*self.airfoil_CLa*np.cos(self.sweep) \
            * (self.V[1]/self.V[0] - self.airfoil_aL0) \
            * (1 - (2.*self.wing_ctrl_xyz[2, :]/self.span)**4)**(1/4)

        if self._aero_approx:
            if jacobian:
                if G0 is None:
                    _G, _info, _ier, _msg = \
                        fsolve(self._LL_func, _guess,
                               xtol=1.e-12, full_output=True,
                               fprime=self._jacobian)
                else:
                    _G, _info, _ier, _msg = \
                        fsolve(self._LL_func, G0, xtol=1.e-12,
                               full_output=True, fprime=self._jacobian)
            else:
```

```
            if G0 is None:
                _G, _info, _ier, _msg = \
                    fsolve(self._LL_func, _guess,
                           xtol=1.e-12, full_output=True)
            else:
                _G, _info, _ier, _msg = \
                    fsolve(self._LL_func, G0, xtol=1.e-12,
                           full_output=True)

        else:
            if G0 is None:
                _G, _info, _ier, _msg = \
                    fsolve(self._LL_func, _guess,
                           xtol=1.e-12, full_output=True)
            else:
                _G, _info, _ier, _msg = \
                    fsolve(self._LL_func, G0, xtol=1.e-12,
                           full_output=True)

        # Catch solver errors
        if _ier is 1:
            _F = 2.*np.sum(np.cross(self._TV_influence @ _G
                                    + self.u[:, None], self._zeta, axis=0)
                           * _G*self.section_area/self.area, axis=1)
            self.circulation_dist = _G*self.Vinf

        else:
            _F = np.array([np.nan, np.nan, np.nan])
            self.circulation_dist = _G*np.nan
            print(_msg)

    if core != 0.0:
        _d = self.wing_ctrl_xyz[2, :, None] - self.wing_xyz[2, None, :-1]
        _w = _d*self.circulation_dist/np.sqrt(core**4 + _d**4)

        _d = self.wing_ctrl_xyz[2, :, None] - self.wing_xyz[2, None, 1:]
        _w -= _d*self.circulation_dist/np.sqrt(core**4 + _d**4)

        _w = -np.sum(_w, axis=1)/(2.*np.pi)

        _trefftz = -np.sum(self.circulation_dist*_w
                           * abs(self.wing_xyz[2, :-1]
                                 - self.wing_xyz[2, 1:])) \
            / (self.Vinf*self.area)

        print('Core:', core)
        print('--->␣Induced␣Drag:', _F[0]*self.u[0] + _F[1]*self.u[1])
        print('--->␣Trefftz␣Drag:', _trefftz)
        print()

    ux, uy, uz = self.u
    _CL = (-_F[0]*uy + _F[1]*ux)/np.sqrt(uy*uy + ux*ux)
    _CD = _F[0]*ux + _F[1]*uy + _F[2]*uz
    _CS = (-_F[0]*ux*uz - _F[1]*uy*uz + _F[2]*(uy*uy + ux*ux)) \
        / np.sqrt(ux*ux*uz*uz + uy*uy*uz*uz + (uy*uy + ux*ux)**2)
```

```
        return _CL, _CD, _CS

    def _aero_properties_calc(self, sweep=0.):
        '''Determines the aerodynamic properties of a swept airfoil.

        Parameters
        ----------
        sweep : scalar, optional
            The local sweep angle of the effective airfoil (rad).

        Returns
        -------
        CLa : scalar
            Effective lift slope of the swept airfoil (1/rad).

        aL0 : scalar
            Effective zero-lift angle of attack of the swept airfoil (rad).
        '''

        _G = [0., 0., 0.]

        _a = np.array([np.radians(-5.), 0., np.radians(5.)])

        _swept_airfoil, _swept_airfoil_ctrl = self._scale_and_sweep(sweep)

        _vpm = VPM(*_swept_airfoil, 0., _swept_airfoil_ctrl)

        _G[0] = _vpm.solve(self._vector_from_angles(_a[0], 0., 1.))[2]

        _G[1] = _vpm.solve(self._vector_from_angles(_a[1], 0., 1.))[2]

        _G[2] = _vpm.solve(self._vector_from_angles(_a[2], 0., 1.))[2]

        _G_mean = (_G[0] + _G[1] + _G[2])/3.

        airfoil_Ga = (_a[0]*(_G[0] - _G_mean)
                      + _a[1]*(_G[1] - _G_mean)
                      + _a[2]*(_G[2] - _G_mean)) \
            / (_a[0]*_a[0] + _a[1]*_a[1] + _a[2]*_a[2])

        airfoil_aL0 = -_G_mean/airfoil_Ga

        return 2.*airfoil_Ga/np.cos(sweep), airfoil_aL0

    def _angles_from_vector(self, V):
        '''Defines flow angles and magnitude from a flow vector.

        Parameters
        ----------
        V : array_like
            An array of size [3] containing the x, y, and z components of the
            free-stream velocity.

        Returns
```

```
        -------
        alpha : scalar
            Angle between the free-stream and the x-z plane (rad).

        beta : scalar
            Angle between the free-stream and the x-y plane (rad).

        Vinf : scalar
            The magnitude of the free-stream velocity.
        '''

        _V0 = V[0]
        _V1 = V[1]
        _V2 = V[2]

        # alpha, beta, Vinf
        return np.arctan2(_V1, _V0), np.arctan2(_V2, _V0), \
            np.sqrt(_V0*_V0 + _V1*_V1 + _V2*_V2)

    def _camber_dy(self, x, c, m, p):
        '''Derivative of the airfoil camber line as a function of x.

        Parameters
        ----------
        x : coordinate at which the derivative is taken.

        c : chord length

        m : maximum camber value

        p : location of the maximum camber

        Returns
        -------
        dyc : derivative of the camber line

        '''

        dyc = np.zeros(len(x))

        dyc[x <= p] = (2.*m)*((1./p) - x[x <= p]/(p*p))
        dyc[x > p] = (-2.*m)*(1./(c - p) - (c - x[x > p])/(c - p)**2)

        return dyc

    def _camber_y(self, x, c, m, p):
        '''Location of the airfoil camber line as a function of x.

        Parameters
        ----------
        x : coordinate at which the derivative is taken.

        c : chord length

        m : maximum camber value
```

```
        p : location of the maximum camber

        Returns
        -------
        yc : Location of the camber line

        '''

        yc = np.zeros(len(x))

        yc[x <= p] = m*(2.*(x[x <= p]/(p)) - (x[x <= p]/(p))**2)
        yc[x > p] = m*(2.*((c - x[x > p])/(c - p))
                       - ((c - x[x > p])/(c - p))**2)

        return yc

    def _df_cond_cav(self, z, z0, c, c_z0, dc, dc_z0, sig, sweep, cr, b, f):
        '''The derivative of the effective LAC, based on Kuchemann's equation.

        Parameters
        ----------
        z : spanwise coordinate

        z0 : control point location

        c : chord length at position z

        c_z0 : chord length at control point z0

        dc : change in chord length at location z, dc/dz

        dc_z0 : change in chord length at control point z0, dc/dz

        sig : blend strength factor

        sweep : global sweep of the wing (rad)

        cr : chord length at the root

        b : span of the wing

        f : LAC model to blend

        Returns
        -------
        x : change in location of the effective aerodynamic center at point z

        '''

        _blend = np.exp(-sig*(z0 - z)**2)

        if f == 'k':
            return self._dkuchemann(z, c, dc, sweep, b) \
                + _blend*(self._dkuchemann(z0, c_z0, dc_z0, sweep, b)
```

```
                    - self._dkuchemann(z, c, dc, sweep, b) - 2*sig*(z0
                                                                    - z)
                    * (self._dkuchemann(z0, c_z0, dc_z0, sweep, b)*(z0
                                                                    - z)
                       - (self._kuchemann(z0, c_z0, cr, sweep, b)
                          - self._kuchemann(z, c, cr, sweep, b))))
    elif f == 'c' or f == 'w':
        return self._dquarter_chord(z, sweep) \
            + _blend*(self._dquarter_chord(z0, sweep)
                    - self._dquarter_chord(z, sweep) - 2*sig*(z0 - z)
                    * (self._dquarter_chord(z0, sweep)*(z0 - z)
                       - (self._quarter_chord(z0, cr, sweep)
                          - self._quarter_chord(z, cr, sweep))))

def _dkuchemann(self, z, c, dc, sweep, b):
    '''Derivative of Kuchemann's model for the LAC.

    Parameters
    ----------
    z : spanwise coordinate

    c : chord length at location z

    dc : change in chord length at location z, dc/dz

    sweep : global sweep of the wing (rad)

    b : span of the wing

    Returns
    -------
    dx : change in the location of the aerodynamic center at location z

    '''

    _sweep_K = sweep/(1. + (self.airfoil_CLa*np.cos(sweep)
                    / (np.pi*self.RA))**2)**(1/4)

    _K = (1. + (self.airfoil_CLa*np.cos(_sweep_K)/(np.pi*self.RA))**2) \
        ** (np.pi/(4.*(np.pi + 2*abs(_sweep_K))))

    if sweep == 0.:

        return -dc*(1. - 1./_K)/4.

    else:

        _tanl = 2*np.pi*np.tan(_sweep_K)/(_sweep_K*c)

        _lam = np.sqrt(1. + (_tanl*z)**2) - _tanl*abs(z)\
            - np.sqrt(1. + (_tanl*(b/2. - abs(z)))**2) + _tanl*(b/2.
                                                                - abs(z))

        _lamp = (_tanl*_tanl*(z*c - z*z*dc)/c)/np.sqrt(1. + (_tanl*z)**2) \
            - _tanl*(np.sign(z)*c - abs(z)*dc)/c \
```

```
            + (_tanl*_tanl*(np.sign(z)*(b/2. - abs(z))*c
                           + dc*(b/2. - abs(z))**2)/c) \
            / np.sqrt(1. + (_tanl*(b/2. - abs(z)))**2) \
            - _tanl*(np.sign(z)*c + (b/2. - abs(z))*dc)/c

        return np.tan(sweep)*np.sign(z) \
            + _lamp*_sweep_K*c/(2.*np.pi*_K) \
            - dc*(1. - (1. + 2.*_lam*_sweep_K/np.pi)/_K)/4.

def _dquarter_chord(self, z, sweep):
    '''Derivative of the quarter-Chord model for the LAC.

    Parameters
    ----------
    z : spanwise coordinate

    cr : chord length at the root

    sweep : global sweep of the wing (rad)

    Returns
    -------
    x : location of the aerodynamic center at location z

    '''

    return np.tan(sweep)*np.sign(z)

def _f_cond_cav(self, z, z0, c, c_z0, dc, dc_z0, sig, sweep, cr, b, f):
    '''The effective LAC, based on Kuchemann's equation.

    Parameters
    ----------
    z : spanwise coordinate

    z0 : control point location

    c : chord length at position z

    c_z0 : chord length at control point z0

    dc : change in chord length at location z, dc/dz

    dc_z0 : change in chord length at control point z0, dc/dz

    sig : blend strength factor

    sweep : global sweep of the wing (rad)

    cr : chord length at the root

    b : span of the wing

    f : LAC model to blend
```

```
        Returns
        -------
        x : location of the effective aerodynamic center at point z

        '''

        _blend = np.exp(-sig*(z0 - z)**2)

        if f == 'k':
            return (1. - _blend)*self._kuchemann(z, c, cr, sweep, b) \
                + _blend*(self._dkuchemann(z0, c_z0, dc_z0, sweep, b)*(z - z0)
                          + self._kuchemann(z0, c_z0, cr, sweep, b))
        elif f == 'c' or f == 'w':
            return (1. - _blend)*self._quarter_chord(z, cr, sweep) \
                + _blend*(self._dquarter_chord(z0, sweep)*(z - z0)
                          + self._quarter_chord(z0, cr, sweep))

    def _jacobian(self, G):
        '''Jacobian of the non-linear, non-dimensional lifting-line equation.

        Parameters
        ----------
        G : array_like
            Circulation distribution normalized by the freestream velocity
            magnitude.

        Returns
        -------
        J : ndarray
            Matrix of partial derivatives.
        '''
        _v = self._TV_influence
        _u = _v @ G + self.u[:, None]

        _ux, _uy, _uz = _u

        _uxz = np.cross(_u, self._zeta, axis=0)

        _uxz_norm = np.linalg.norm(_uxz, axis=0)

        _vxz = np.cross(_v, self._zeta[:, :, None], axis=0)

        J = 2.*(_uxz[0, :, None]*_vxz[0, :, :] + _uxz[1, :, None]*_vxz[1, :, :]
                + _uxz[2, :, None]*_vxz[2, :, :]) \
            * G[:, None]/_uxz_norm[:, None]

        J += 2.*np.diag(_uxz_norm)

        _CLa, _aL0 = self._aero_properties

        _sweep = self.local_sweep_ctrl
        _Cs = np.cos(_sweep)
        _Ss = np.sin(_sweep)

        _a = np.arctan2(_uy, _ux)
```

```
        _b = np.arctan2(_uz, _ux)
        _aL = np.arctan2(_uy, _ux*_Cs + _uz*_Ss)
        _bL = _b - _sweep

        _da = (_ux[:, None]*_v[1, :, :] - _uy[:, None]*_v[0, :, :]) \
            / (_ux[:, None]*_ux[:, None] + _uy[:, None]*_uy[:, None])
        _db = (_ux[:, None]*_v[2, :, :] - _uz[:, None]*_v[0, :, :]) \
            / (_ux[:, None]*_ux[:, None] + _uz[:, None]*_uz[:, None])
        _daL = ((_ux[:, None]*_Cs[:, None]
                 + _uz[:, None]*_Ss[:, None])*_v[1, :, :]
                - _uy[:, None]*(_v[0, :, :]*_Cs[:, None]
                                + _v[2, :, :]*_Ss[:, None])) \
            / (_ux[:, None]*_ux[:, None]
               + (_ux[:, None]*_Cs[:, None] + _uz[:, None]*_Ss[:, None])**2)

        _Ca = np.cos(_a)
        _Sa = np.sin(_a)
        _Cb = np.cos(_b)
        _Sb = np.sin(_b)
        _CaL = np.cos(_aL)
        _SaL = np.sin(_aL)
        _CbL = np.cos(_bL)
        _SbL = np.sin(_bL)

        _Rn = np.sqrt(_Ca*_Ca*_CbL*_CbL + _Sa*_Sa*_Cb*_Cb)
        _Rd = np.sqrt(1. - _Sa*_Sa*_Sb*_Sb)
        _RLd = np.sqrt(1. - _SaL*_SaL*_SbL*_SbL)

        _R = _Rn/_Rd
        _RL = _CbL/_RLd

        _dR = (_Sa*_Ca*(_Sb*_Sb*_Rn/(_Rd*_Rd)
                        + (_Cb*_Cb - _CbL*_CbL)/_Rn)/_Rd)[:, None]*_da \
            + ((_Sa*_Sa*_Sb*_Cb*_Rn/(_Rd*_Rd)
                - (_Ca*_Ca*_SbL*_CbL + _Sa*_Sa*_Sb*_Cb)/_Rn)/_Rd)[:, None]*_db

        _dRL = (_SaL*_CaL*_SbL*_SbL*_CbL/(_RLd*_RLd*_RLd))[:, None]*_daL \
            - (_CaL*_CaL*_SbL/(_RLd*_RLd*_RLd))[:, None]*_db

        _dCL = _dR*_RL[:, None]*_CLa[:, None]*(_aL[:, None] - _aL0[:, None]) \
            + _R[:, None]*_dRL*_CLa[:, None]*(_aL[:, None] - _aL0[:, None]) \
            + _R[:, None]*_RL[:, None]*_CLa[:, None]*_daL

        J -= _dCL

        return J

    def _kuchemann(self, z, c, cr, sweep, b):
        '''Kuchemann's model for the LAC.

        Parameters
        ----------
        z : spanwise coordinate

        c : chord length at location z
```

```
        cr : chord length at the root

        sweep : global sweep of the wing (rad)

        b : span of the wing

        Returns
        -------
        x : location of the aerodynamic center at location z

        '''

        _sweep_K = sweep/(1. + (self.airfoil_CLa*np.cos(sweep)
                          / (np.pi*self.RA))**2)**(1/4)

        _K = (1. + (self.airfoil_CLa*np.cos(_sweep_K)/(np.pi*self.RA))**2) \
            ** (np.pi/(4.*(np.pi + 2*abs(_sweep_K))))

        if sweep == 0.:

            return 0.25*cr - c*(1. - 1./_K)/4.

        else:
            _tanl = 2*np.pi*np.tan(_sweep_K)/(_sweep_K*c)

            _lam = np.sqrt(1. + (_tanl*z)**2) - _tanl*abs(z)\
                - np.sqrt(1. + (_tanl*(b/2. - abs(z)))**2) + _tanl*(b/2.
                                                                    - abs(z))

            return 0.25*cr + np.tan(sweep)*abs(z) \
                - c*(1. - (1. + 2.*_lam*_sweep_K/np.pi)/_K)/4.

    def _lift_from_aero(self, CLa, aL0, sweep, V, Vref=1.):
        '''Calculates the lift coefficient of a swept airfoil.

        Parameters
        ----------
        CLa : scalar
            The effective lift slope of the swept airfoil.

        aL0 : scalar
            The efefctive zero-lift angle of attack of the swept airfoil (rad).

        sweep : scalar
            The local sweep angle of the effective airfoil (rad).

        V : array_like
            An array of size [3] containing the x, y, and z components of the
            local velocity.

        Vref : scalar, optional
            The magnitude of the freestream (reference) velocity.
        '''
        if self.thin_approx:
```

```
            _a, _b, _V = self._angles_from_vector(V)

            return CLa*(_a - aL0)*(np.cos(_b) + np.sin(_b)*np.tan(sweep))

        else:
            _a_eff, _b_eff, _Vn = self._sweep_vector(sweep, V)

            _beta_fact = np.cos(_b_eff)/(np.sqrt(1. - np.sin(_a_eff)**2
                                                 * np.sin(_b_eff)**2))

            return CLa*(_a_eff - aL0)*_beta_fact*_Vn/Vref

    def _LL_func(self, G):
        '''Non-linear, non-dimensional lifting-line equation.

        Parameters
        ----------
        G : array_like
            Circulation distribution normalized by the freestream velocity
            magnitude.

        Returns
        -------
        R : array_like
            Array of the residuals between the lift values predicted from
            section properties and from circulation.
        '''

        _Vi = self._TV_influence @ G + self.u[:, None]

        if self._aero_approx:
            _CL = self._lift_from_aero(*self._aero_properties,
                                       self.local_sweep_ctrl, self.Vinf*_Vi,
                                       self.Vinf)
        else:
            _CL = np.array([vpm.solve(V_local, Vref=self.Vinf)[0]
                            for vpm, V_local in
                            zip(self._vortex_panel_method_objects,
                                self.Vinf*_Vi.T)])

        _dF = 2.*np.linalg.norm(np.cross(_Vi, self._zeta, axis=0), axis=0)*G

        return _dF - _CL

    def _NACA_4(self, node_count, NACA, cluster, cp_cluster, openTE):
        '''The definition of an airfoil in the NACA 4-digit series (XXXX).

        Parameters
        ----------
        node_count : int
            Total number of control points around the airfoil.

        NACA : str
            String containing the NACA 4-digit airfoil designation ('XXXX').
```

```
cluster : int
    Defines the node distribution (0 = even distribution,
                                   1 = cluster at LE and TE).

cp_cluster : int
    Defines the control point distribution (0 = mid-point of panel,
                                            1 = cluster at LE and TE).
openTE : bool
    Flag indicating if the trailing edge should be closed (False) or
    open (True).
'''

if node_count % 2 == 0:
    print("<Airfoil control point count rounded down to odd number:",
          node_count-1, ">")
else:
    node_count += 1

_chord = 1.  # Assuming a chord of one allows for easy scaling
self.airfoil_n = node_count
self.airfoil_name = NACA
self.airfoil_xy = np.zeros((2, node_count))
self.airfoil_ctrl_xy = np.zeros((2, node_count-1))

_max_camber = float(NACA[0])*_chord/100.
_max_camber_loc = float(NACA[1])*_chord/10.
_max_thickness = float(NACA[2:])*_chord/100.

_theta = np.linspace(0, 2.*np.pi, node_count)
if cluster == 1:
    _camber_x = .5*_chord*(1 + np.cos(_theta))
elif cluster == 0:
    _camber_x = _chord*abs(1 - _theta/(np.pi))

if _max_camber == 0.:
    self.airfoil_xy[0, :] = _camber_x
    self.airfoil_xy[1, :] = np.sign(_theta - np.pi) \
        * self._thickness(_camber_x, _chord, _max_thickness, openTE)

    if self._weiss:
        self._cam_norm = np.array([0., 1., 0.])

else:
    _camber_angle = np.arctan(self._camber_dy(_camber_x, _chord,
                              _max_camber,
                              _max_camber_loc))
    self.airfoil_xy[0, :] = _camber_x \
        + np.sign(np.pi - _theta) \
        * self._thickness(_camber_x, _chord, _max_thickness, openTE) \
        * np.sin(_camber_angle)
    self.airfoil_xy[1, :] = \
        self._camber_y(_camber_x, _chord, _max_camber,
                       _max_camber_loc)\
        + np.sign(_theta - np.pi) \
        * self._thickness(_camber_x, _chord, _max_thickness, openTE) \
```

```
            * np.cos(_camber_angle)

        if self._weiss:
            _dydx_34 = self._camber_dy(np.array([0.75]), 1., _max_camber,
                                       _max_camber_loc)[0]
            self._cam_norm = np.array([-_dydx_34, 1., 0.]) \
                / np.sqrt(_dydx_34*_dydx_34 + 1.)

    if cp_cluster == 1:

        _theta = np.linspace(0, 2.*np.pi, 2*node_count-1)[1::2]
        if cluster == 1:
            _camber_x = .5*_chord*(1 + np.cos(_theta))
        elif cluster == 0:
            _camber_x = _chord*abs(1 - _theta/(np.pi))

        if _max_camber == 0.:
            self.airfoil_ctrl_xy[0, :] = _camber_x
            self.airfoil_ctrl_xy[1, :] = np.sign(_theta - np.pi) \
                * self._thickness(_camber_x, _chord, _max_thickness,
                                  openTE)

        else:
            _camber_angle = np.arctan(self._camber_dy(_camber_x, _chord,
                                      _max_camber,
                                      _max_camber_loc))
            self.airfoil_ctrl_xy[0, :] = _camber_x \
                + np.sign(np.pi - _theta) \
                * self._thickness(_camber_x, _chord, _max_thickness,
                                  openTE) \
                * np.sin(_camber_angle)
            self.airfoil_ctrl_xy[1, :] = \
                self._camber_y(_camber_x, _chord, _max_camber,
                               _max_camber_loc)\
                + np.sign(_theta - np.pi) \
                * self._thickness(_camber_x, _chord, _max_thickness,
                                  openTE) \
                * np.cos(_camber_angle)
    else:
        self.airfoil_ctrl_xy = (self.airfoil_xy[:, 1:]
                                + self.airfoil_xy[:, :-1])/2.

def _normalize(self, V):
    '''Normalizes a flow vector to have a magnitude of unity.

    Parameters
    ----------
    V : array_like
        An array of size [3] containing the x, y, and z components of the
        free-stream velocity.

    Returns
    -------
    V_norm : array_like
        An array of size [3] containing the x, y, and z components of the
```

```
            normalized free-stream velocity.

        '''
        _V1, _V2, _V3 = V

        _nsqrt = 1/np.sqrt(_V1*_V1 + _V2*_V2 + _V3*_V3)

        return np.array([_V1*_nsqrt, _V2*_nsqrt, _V3*_nsqrt])

    def _quarter_chord(self, z, cr, sweep):
        '''Quarter-Chord model for the LAC.

        Parameters
        ----------
        z : spanwise coordinate

        cr : chord length at the root

        sweep : global sweep of the wing (rad)

        Returns
        -------
        x : location of the aerodynamic center at location z

        '''

        return 0.25*cr + np.tan(sweep)*abs(z)

    def _scale_and_sweep(self, angle=0., factor=1.):
        '''The effective swept airfoil geometry.

        Parameters
        ----------
        angle : scalar, optional
            The sweep angle at which the effective airfoil coordinates will be
            calculated (rad).

        factor : scalar, optional
            The scaling by which the coordinates are multiplied to match the
            desired chord length.

        '''

        return factor*np.array([self.airfoil_xy[0, :]*np.cos(angle),
                                self.airfoil_xy[1, :]]), \
            factor*np.array([self.airfoil_ctrl_xy[0, :]*np.cos(angle),
                             self.airfoil_ctrl_xy[1, :]])

    def _straight_segment(self, start, end, point, v_core):
        '''Influence of a straight vortex segment on a point.

        Parameters
        ----------
        start : array_like
            The position vector of the beginning point of the vortex segment,
```

```
            in three dimensions.
        end : array_like
            The position vector of the end point of the vortex segment,
            in three dimensions.
        point : array_like
            The position vector of the point at which the influence of the
            vortex segment is calculated, in three dimensions.
        v_core : scalar
            The radius of the vortex finite core.

        Returns
        -------
        influence : array_like
            The influence of vortex segment at the point, in three dimensions.

        '''

        _point_array = np.array(point, dtype=float)
        _r1 = _point_array - np.array(start, dtype=float)
        _r2 = _point_array - np.array(end, dtype=float)

        _r1_mag = np.linalg.norm(_r1, axis=0)
        _r2_mag = np.linalg.norm(_r2, axis=0)

        _r1_r2 = _r1 - _r2
        _r1_r2_mag = np.linalg.norm(_r1_r2, axis=0)
        _r1dotr2 = _r1[0]*_r2[0] + _r1[1]*_r2[1] + _r1[2]*_r2[2]
        _r1dotr1_r2 = _r1[0]*_r1_r2[0] + _r1[1]*_r1_r2[1] + _r1[2]*_r1_r2[2]
        _r2dotr1_r2 = _r1_r2[0]*_r2[0] + _r1_r2[1]*_r2[1] + _r1_r2[2]*_r2[2]

        with np.errstate(divide='ignore', invalid='ignore'):

            d = np.linalg.norm(np.cross(_r1, _r1_r2, axis=0), axis=0)\
                / _r1_r2_mag
            d = np.where(_r1dotr1_r2 < 0., _r1_mag, d)
            d = np.where(_r2dotr1_r2 > 0., _r2_mag, d)

            influence = (_r1_mag + _r2_mag)*np.cross(_r1, _r2, axis=0)
            influence *= d*d/np.sqrt(v_core*v_core*v_core*v_core + d*d*d*d)
            influence /= 4.*np.pi*_r1_mag*_r2_mag*(_r1_mag*_r2_mag + _r1dotr2)

        return np.nan_to_num(influence, True, 0.0, 0.0, 0.0)

    def _straight_semi_infinite(self, start, end_vec, point, v_core):
        '''Influence of a straight semi-infinite vortex on a point.

        Parameters
        ----------
        start : array_like
            The position vector of the beginning point of the semi-infinite
            vortex, in three dimensions.
        end_vec : array_like
            The unit vector pointing from the beginning point of the
            semi-infinite vortex to the infinity, in three dimensions.
        point : array_like
```

```
            The position vector of the point at which the influence of the
            semi-infinite vortex is calculated, in three dimensions.
        v_core : scalar
            The radius of the vortex finite core.

        Returns
        -------
        influence : array_like
            The influence of vortex segment at the point, in three dimensions.

        '''

        _r1 = np.array(point, dtype=float) - np.array(start, dtype=float)
        _u_inf = np.array(end_vec, dtype=float)

        _r1_mag = np.linalg.norm(_r1, axis=0)
        _r1dotu_inf = _r1[0]*_u_inf[0] + _r1[1]*_u_inf[1] + _r1[2]*_u_inf[2]

        d = np.linalg.norm(np.cross(_r1, _u_inf, axis=0), axis=0)
        d = np.where(_r1dotu_inf < 0., _r1_mag, d)

        influence = np.cross(_u_inf, _r1, axis=0)
        influence *= d*d/np.sqrt(v_core*v_core*v_core*v_core + d*d*d*d)
        influence /= 4.*np.pi*_r1_mag*(_r1_mag - _r1dotu_inf)

        return np.nan_to_num(influence, True, 0.0, 0.0, 0.0)

    def _sweep_angles(self, angle, alpha=0., beta=0., Vinf=1.):
        '''Effective 2D flow properties for a sweep angle from flow angles.

        Parameters
        ----------
        angle : scalar
            The sweep angle at which the effective airfoil coordinates will be
            calculated (rad).

        alpha : scalar, optional
            Angle between the free-stream and the x-z plane (rad).

        beta : scalar, optional
            Angle between the free-stream and the x-y plane (rad).

        Vinf : scalar, optional
            The magnitude of the free-stream velocity.

        Returns
        -------
        alpha : scalar
            Effective angle of attack.

        Vinf : scalar
            Effective free-stream velocity.

        '''
```

```
        _Ca = np.cos(alpha)
        _Sa = np.sin(alpha)
        _Cb = np.cos(beta)
        _Sb = np.sin(beta)
        _sqrtSaSb = np.sqrt(1. - _Sa*_Sa*_Sb*_Sb)

        alpha = np.arctan2(np.tan(alpha)*_Cb,
                           np.cos(angle - _beta))

        Vinf = Vinf*np.sqrt(_Ca*_Ca*np.cos(angle - beta)**2
                            + _Sa*_Sa*_Cb*_Cb)/_sqrtSaSb

        return alpha, _beta - angle, Vinf

    def _sweep_vector(self, angle, V):
        '''Effective 2D flow properties for a sweep angle from flow vector.

        Parameters
        ----------
        angle : scalar
            The sweep angle at which the effective airfoil coordinates will be
            calculated (in radians).

        V : array_like
            An array of size [3] containing the x, y, and z components of the
            free-stream velocity.

        Returns
        -------
        alpha : scalar
            Effective angle of attack.

        Vinf : scalar
            Effective free-stream velocity.

        '''
        _alpha, _beta, _Vinf = self._angles_from_vector(V)

        _Ca = np.cos(_alpha)
        _Sa = np.sin(_alpha)
        _Cb = np.cos(_beta)
        _Sb = np.sin(_beta)
        _sqrtSaSb = np.sqrt(1. - _Sa*_Sa*_Sb*_Sb)

        alpha = np.arctan2(np.tan(_alpha)*_Cb,
                           np.cos(angle - _beta))

        Vinf = _Vinf*np.sqrt(_Ca*_Ca*np.cos(angle - _beta)**2
                             + _Sa*_Sa*_Cb*_Cb)/_sqrtSaSb

        return alpha, _beta - angle, Vinf

    def _thickness(self, x, c, t, openTE):
        '''Thickness of the airfoil as a function of x.
```

```
        Parameters
        ----------
        x : coordinate at which the derivative is taken.

        c : chord length

        t : maximum thickness of the airfoil

        openTE : flag defining trailing edge closure

        Returns
        -------

        tx : airfoil thickness at point x

        '''
        if openTE:
            return .5*t*(2.969*np.sqrt(x/c) - 1.260*(x/c) - 3.516*((x/c)**2)
                         + 2.843*((x/c)**3) - 1.015*((x/c)**4))
        else:
            return .5*t*(2.980*np.sqrt(x/c) - 1.320*(x/c) - 3.286*((x/c)**2)
                         + 2.441*((x/c)**3) - 0.815*((x/c)**4))

    def _vector_from_angles(self, alpha=0., beta=0., Vinf=1.):
        '''Defines a flow vector from flow angles and magnitude.

        Parameters
        ----------
        alpha : scalar, optional
            Angle between the free-stream and the x-z plane (rad).

        beta : scalar, optional
            Angle between the free-stream and the x-y plane (rad).

        Vinf : scalar, optional
            The magnitude of the free-stream velocity.

        Returns
        -------
        V : array_like
            An array of size [3] containing the x, y, and z components of the
            free-stream velocity.
        '''

        _Ca = np.cos(alpha)
        _Sa = np.sin(alpha)
        _Cb = np.cos(beta)
        _Sb = np.sin(beta)
        _sqrtSaSb = np.sqrt(1. - _Sa*_Sa*_Sb*_Sb)

        return Vinf*np.array([_Ca*_Cb,
                              _Sa*_Cb,
                              _Ca*_Sb])/_sqrtSaSb
```

CURRICULUM VITAE

Jackson T. Reid

EDUCATION

Ph.D. Aerospace Engineering **May 2020**
Utah State University (USU), Logan, UT **GPA 4.00**
Dissertation: A General Approach to Lifting-Line Theory,
Applied to Wings with Sweep

M.S. Mechanical Engineering **May 2020**
Utah State University, Logan, UT **GPA 4.00**
Coursework in: Aerodynamics, Aircraft Simulation, Astrodynamics,
CFD, Control Systems, Dynamics/Vibrations,
Fluid Dynamics, Optimization, Propulsion Systems,
and Unmanned Aerial Systems

B.S. Mechanical Engineering **May 2016**
Utah State University, Logan, UT **GPA 3.88**
Emphasis: Aerospace Engineering
Minor: Mathematics
Honors: Magna Cum Laude

JOURNAL ARTICLES

Reid, Jackson T. and Hunsaker, Douglas F., **"A General Approach to Lifting-Line Theory, Applied to Wings with Sweep"**, [Manuscript Under Review]

Malmendier, Andreas and *Reid, Jackson T.*, **"Application of the Biot-Savart Law to Parabolic Vortex Segments using Elliptic Integrals"**, Zeitschrift für Angewandte Mathematik und Physik, Accepted, arXiv:1907.10131

Hunsaker, Douglas F., *Reid, Jackson T.*, and Joo, James J., **"Geometric Definition and Ideal Aerodynamic Performance of Parabolic Trailing-Edge Flaps"**, International Journal of Astronautics and Aeronautical Engineering, Vol. 4, No. 1, March 2019, doi: 10.35840/2631-5009/7526

CONFERENCE PAPERS

Reid, Jackson T. and Hunsaker, Douglas F., **"A General Approach to Lifting-Line Theory, Applied to Wings with Sweep"**, AIAA SciTech Forum, January 2020, doi: 10.2514/6.2020-1287

Reid, Jackson T. and Hunsaker, Douglas F., **"Effect of Sweep on Airfoil Section Properties"**, AIAA SciTech Forum, January 2019, doi: 10.2514/6.2019-2118

Hunsaker, Douglas F., *Reid, Jackson T.*, Moorthamers, Bruno, and Joo, James J., **"Geometric Definition and Ideal Aerodynamic Performance of Parabolic Trailing-Edge Flaps"**, AIAA SciTech Forum, January 2018, doi: 10.2514/6.2018-1278

Reid, Jackson T. and Hunsaker, Douglas F., **"Implementation of OpenFOAM for Inviscid, Incompressible Aerodynamic Flows"**, Fellowship Symposium of the Utah NASA Space Grant Consortium, May 2017

PRESENTATIONS

AIAA SciTech Forum, AIAA, 2018, 2019, and 2020
USU Student Research Symposium, USU, 2018 and 2020
Fellowship Symposium of the Utah NASA Space Grant Consortium, NASA, 2017

AWARDS

Outstanding PhD Researcher, USU MAE Department, 2020
Presidential Doctoral Research Fellowship, USU, 2016–2020